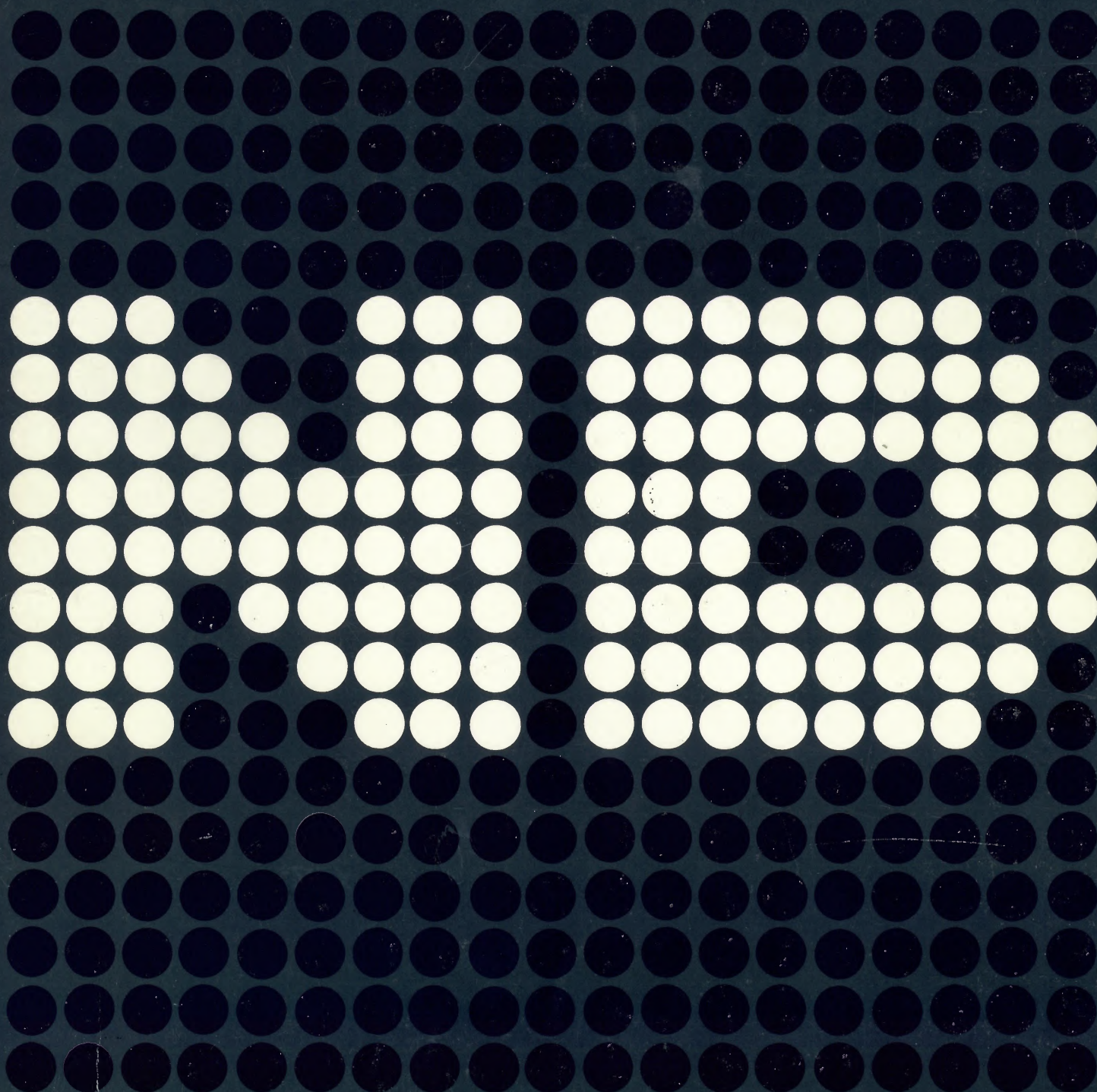


NORD-10/S
MICROPROGRAM

NORSK DATA A.S



NORD-10/S Microprogram
Publication No. ND-06.010.01

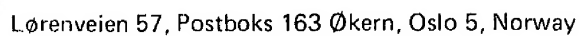


TABLE OF CONTENTS

+ + +

<i>Section:</i>	<i>Page:</i>
1 INTRODUCTION	1-1
1.1 Philosophy of Microprogramming	1-1
1.2 Micro Processor Instruction Set	1-2
1.3 Microprogram Control	1-2
1.4 Entry Point Generator	1-4
1.5 The OR Logic	1-5
2 MICROINSTRUCTION DESCRIPTION	2-1
2.1 The Arithmetic Microinstruction	2-1
2.2 The Interblock Microinstruction	2-9
2.3 The Jump Microinstruction	2-11
2.4 The Loop Microinstruction	2-13
3 THE MICROPROGRAM	3-1
3.1 MICMAC – Micro MAC Mnemonic Table	3-1
3.2 NORD-10 Instructions and their Corresponding Entry-Points	3-8
3.2.1 Special Entry Points	3-9
3.3 Labels Referenced in NORD-10 Microprogram	3-10
3.4 The Microprogram Listing	3-11

Figures:

1.1 Micro Instruction's Bit Assignment	1-3
1.2 CPU Control Section	1-7
2.1 Arithmetic μ instruction – Bit Assignment I	2-2
2.2 Arithmetic μ instruction – Bit Assignment II	2-4
2.3 Special Case 1	2-5
2.4 Special Case 1 – Illustration	2-6
2.5 Special Case 2	2-7
2.6 Special Case 3	2-8
2.7 Interblock μ instruction – Bit Assignment	2-10
2.8 Jump μ instruction – Bit Assignment	2-12
2.9 Loop μ instruction – Bit Assignment I	2-14
2.10 Loop μ instruction – Bit Assignment II	2-15

READERS – Please Note!!!!

We frequently refer to the NORD computer in this manual as "NORD-10". However, this **does not** mean that it only applies to NORD-10 users. Please note that it **also** applies to NORD-10/S, NORD-12 and NORD-42 users. We have written "NORD-10" merely for convenience sake.

1 INTRODUCTION

The microprogram is designed to implement, in hardware, the instruction set of NORD-10, NORD-42 and NORD-10/S.

The microprocessor instruction set consists of four micro-instructions.

This manual describes the exact format of the four different micro-instructions together with some examples of usage.

The micro-instructions are stored in a 1k x 32 bits Read Only Memory – ROM.

Chapter 7 contains a listing of the μ -program.

The ROM is logically divided into the following sections:

- μ -programmed execution of NORD's 10/S, 10, 42 instruction repertoire
- μ -programmed operator panel driver
- μ -programmed operator communication in stop mode MOPC
- μ -programmed bootstrap loader
- μ -programmed memory check

1.1 PHILOSOPHY OF MICROPROGRAMMING

Microprogramming is primarily an orderly and systematic means of implementing control logic. By using microprogrammed control, the CPU control section may be broken down into well-defined subsections. This approach simplifies design, documentation and testing.

Flexibility is an advantage of microprogramming: new instructions may be added without changing hardware design or test methods. Alternate instruction sets are available: at present one of two floating point forms may be ordered; 32 bit or 48 bit.

1.2

MICRO PROCESSOR INSTRUCTION SET

The microprocessor instruction set consists of four instructions. These are ARITHMETIC, INTERBLOCK, JUMP and LOOP. This chapter deals with the exact format of these four instructions together with examples on how they may be used.

The operation code is contained in bits 30 and 31 in the Read Only Memory – ROM.

ROM 31	ROM 30	Instruction
0	0	ARITHMETIC
0	1	INTERBLOCK
1	0	JUMP
1	1	LOOP

Refer to Figure 1.1. The format shown applies to Read Only Memory and not Microinstruction Register – MIR. The two are not necessarily identical, due to the function of the OR logic.

The four instructions will be described in the following figure.

1.3

MICROPROGRAM CONTROL

The CPU control logic transforms the content of the instruction register (IR) into a sequence of actions on CPU registers, memory and/or I/O system. These actions are controlled by a set of control signals to registers, selectors, arithmetic elements, memory, I/O system, etc. In a non-microprogrammed machine, these signals are derived directly from the instruction register and a large and complicated Time Counter/Cycle Counter. This type of control logic is not easily structured and is difficult to describe and understand.

A block diagram of the transformation from machine instructions (IR) into a sequence of microinstructions is shown in Figure 1.2. Each NORD machine instruction is executed by a sequence of one or more microinstructions, a microprogram routine.

ARITHMETIC:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP		ALU				AR SEL	CYCLE	C H L E V	S S A V E	OR S P E C S	C O N D	TC		DEST				B				A									
0 0											BIT NO.																				

INTERBLOCK:

31 30		29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP		ALU				ARSEL		CYCLE		DIRECT		SAVE		OR SPECS		LEVEL				DEST				B				A			
0 1						SEL				ECT		E																			

JUMP:

31 30		29 28		27 26		25 24		23 22		21 20		19 18		17 16		15 14		13 12		11 10		9 8		7 6		5 4		3 2		1 0	
OP		C	P	R	I	V	0				0 0 0 0 0 0 0 0				C	O	N	D	T	C	ADDRESS (ABSOLUTE)										
1 0							R																								

LOOP:

31 30		29 28		27 26		25 24		23 22		21 20		19 18		17 16		15 14		13 12		11 10		9 8		7 6		5 4		3 2		1 0							
OP		ALU				ALT ALU				SSAVE		ORSH		0 0		SHR		SHIFT TYPE		SH32		0		SETM		TG CONT		B				TERM		DIVINP		ALTTSP	
1 1																																					

Figure 1.1: Micro Instruction's Bit Assignment

The microprogram entry point is generated from the machine instruction operation code by hardware. This corresponds to the instruction decoding in a non-microprogrammed machine. The microprogram is controlled by a microprogram counter, which points to the next microinstruction to be executed from the Read Only microprogram memory (ROM). Branching may be done by the microinstruction JUMP. The microinstruction counter may be read, thus providing a simple subroutine capability. ROM word length is 32 bits. ROM content is clocked into the microinstruction register, MIR, at the end of each microinstruction. The microword contains information to establish the setting of the control lines for each cycle.

Since the microword (32 bits) is not sufficient to establish the setting of all the control lines, the microword is divided into four groups or instructions, given by ROM bits 31 and 30. The remainder of the 30 bits are in some of the instructions divided into fields having the same meaning in different instructions.

The microinstruction format is tailored to the CPU structure, while keeping the target instruction set (NORD-10) in mind in order to maintain execution efficiency. Many control signals are taken directly from MIR outputs, while others are derived by simple logic from MIR bits and a small Time Counter.

1.4

ENTRY POINT GENERATOR

Refer to Figure 1.2 for the following discussion.

A microprogram terminates by fetching the next machine instruction to be executed. The instruction is placed in the instruction register (IR). The Entry Point Generator (EPG) will then generate a unique address (Entry Point) based on the content of the instruction register (IR). This address will be clocked into the microprogram counter (MPC).

Entry points for all NORD-10 instructions which do not have sub-instruction fields, are 100_8 , 102_8 , . . . , 172_8 for operation codes 0, 1, . . . , 35_8 , respectively; i.e., the Entry Point equals $100 + (\text{operation code}) \cdot 2$.

Example:

LDA — opcode (bits 11, 12, 13, 14 and 15) = $01001 = 11_8$

Entry point = $100 + 11_8 \cdot 2 = 122_8$

In location 122_8 in ROM, resides the first (of two) microinstructions which constitutes the microprogram for the LDA instruction.

A spacing of two locations between the Entry Points is chosen, due to the fact that most of these microprograms occupy two locations of ROM. This applies to the instructions:

LDA, LDX, LDT,
STA, STX, STT, STZ,
JMP, JPL.

Some other instructions, such as FSB, FAB, FMU, FDV, STF, etc., require more than two locations, but a jump to another address in the ROM where the rest of the microprogram for relevant instruction resides is executed.

For all other instructions, the Entry Point is generated according to a spacing of 16 between the EP's for the main instruction operation code such as CJP, ROP, etc. Refer to table "EP for Instructions with Sub-instructions".

Section 3.2 gives the entry points for the NORD-10 instructions.

1.5

THE OR LOGIC

The instruction set for the NORD-10 may be divided into two main groups:

1. Instructions well defined by the operation code (upper 5 bits), will not require an OR logic to be implemented.

Example:

LDA, STA, ADD, AAA, SAA.

2. Instructions not completely defined by the operation code are defined by their subinstruction field. The subinstruction field will give additional information to the operation code.

Example:

- The subinstruction field of a SHIFT instruction will give information about shift direction and shift method.
- The subinstruction field of a SKIP instruction will give information about a skip condition.

To reduce the number of Entry Points in the ROM (not having one for each combination of subinstruction field) the OR logic is introduced to take information directly from the subinstruction field in the Instruction Register (IR) to the Micro-Instruction Register (MIR). The ROM bits 16, 17 and 18 (8 combinations) decide which IR bits are to be transferred to MIR.

The subinstruction field (giving the large instruction repertoire combinations) gives, by means of the OR logic, the microprocessor the necessary information through a minimum of logic.

From Figure 1.2 we can see that micro-instruction register (MIR) bits 0-15 is the output from the OR logic. Table 1.1 describes the origin of the MIR 0-15 for the 8 different OR specifications.

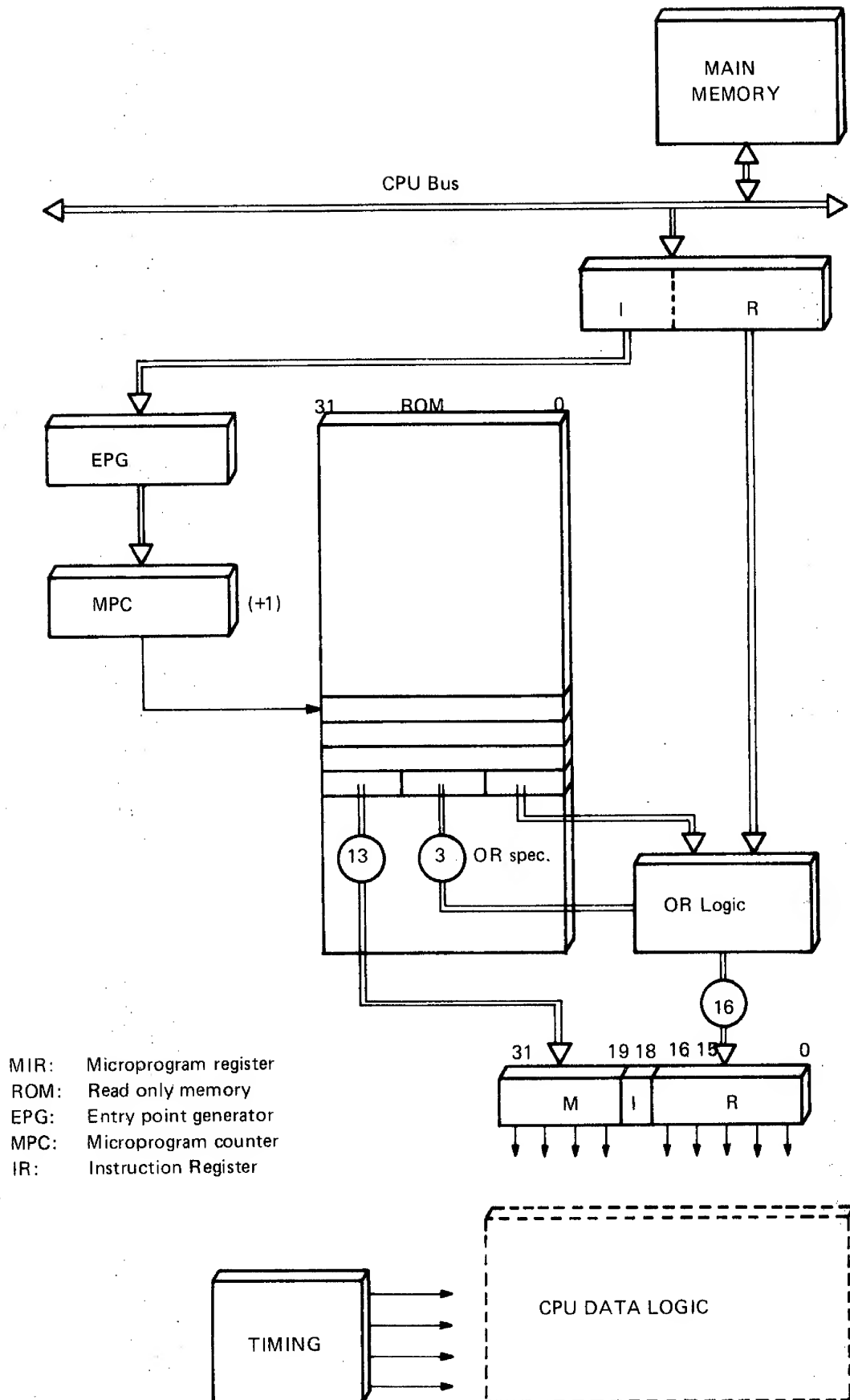


Figure 1.2: CPU Control Section

OR Specifications				MIR 0-15																Comments
No.	Name	MNE	Instr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	NO OR	—	—	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	—
1	OR for BOP without Dest.	ARBWO	ARIT	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	0	I2	I1	I0	If I0-2 ≠ 0
		ARBWO	ARIT	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	1	0	0	0	If I0-2 = 0
		ARBWO	INTB	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	0	I2	I1	I0	If I0-2 ≠ 0
		ARBWO	INTB	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	1	0	0	0	If I0-2 = 0
		ARBWO	INTB	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	1	1	1	0	If I0-2 = 2
3	OR for BOP with Dest.	ARBWO	ARIT	I6	I5	I4	I3	0	I2	I1	I0	R7	R6	R5	R4	0	I2	I1	I0	If I0-2 ≠ 0
		ARBWO	ARIT	I6	I5	I4	I3	1	0	0	0	R7	R6	R5	R4	1	0	0	0	If I0-2 = 0
		ARBWO	INTB	I6	I5	I4	I3	0	I2	I1	I0	R7	R6	R5	R4	R3	R2	R1	R0	If I0-2 ≠ 0
		ARBWO	INTB	I6	I5	I4	I3	1	1	1	0	R7	R6	R5	R4	R3	R2	R1	R0	If I0-2 = 0
		ARBWO	INTB	I6	I5	I4	I3	1	1	1	0	R7	R6	R5	R4	R3	R2	R1	R0	If I0-2 = 2
4	OR for SKP, SHT	ORSHT	ARIT	R15	I10	I9	I8	R11	R10	R9	R8	R7	I2	I1	I0	0	I5	I4	I3	If not COND.
		ORSHT	ARIT	R15	I10	I9	I8	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	If COND.
		ORSHT	LOOP	I5	I10	I9	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	—
5	OR for ROP	ORROP	ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	I2	I1	I0	0	I5	I4	I3	If I6 = 0
		ORROP	ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	0	0	0	0	I5	I4	I3	If I6 = 1
6	OR for SWAP cycle 2	ORSW2	ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	I2	I1	I0	0	I5	I4	I3	If not COND.
		ORSW2	ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	R6	R5	R4	R3	R2	R1	R0	If COND.
7	OR for SWAP cycle 3	ORSW3	ARIT	R15	R14	R13	R12	0	I5	I4	I3	R7	I2	I1	I0	0	I5	I4	I3	—

ARIT = ARITHM

INTB = INTERBLOCK

Ixx = IRxx

Rxx = ROMxx

COND. = CONDITIONAL

Table 1.1: OR Specifications

2 MICROINSTRUCTION DESCRIPTION

2.1 *THE ARITHMETIC MICROINSTRUCTION*

This is the most frequently used μ -instruction in the microprogram. It is used to perform arithmetical as well as logical operations. Refer to Table 2.1.

This μ -instruction is used to set up memory communication (cycle specifications).

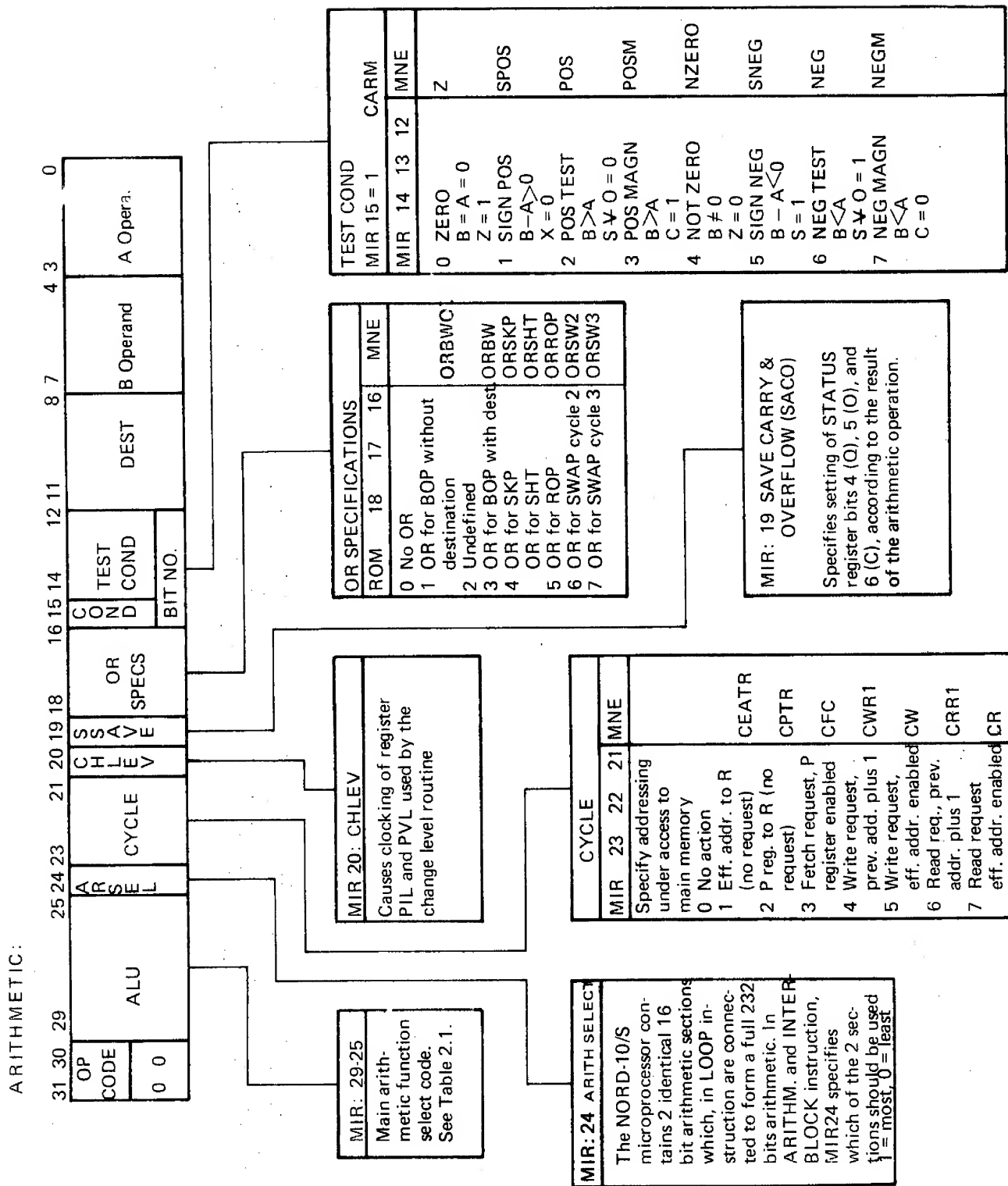
For communication with the internal registers three special cases of the instruction exist. Those cases are illustrated in Figures 2.3, 2.4, 2.5, and 2.6.

If bit 15 in an ARITHMETIC instruction is set, the execution of the microinstruction is dependent on the result of a specified test.

The CARM instruction is a conditional ARITHMETIC instruction using the most significant unit.

For a CARM instruction, the arithmetical or logical operation specified will not be executed if the result of the specified test is false. Any cycle specification will, however, be executed.

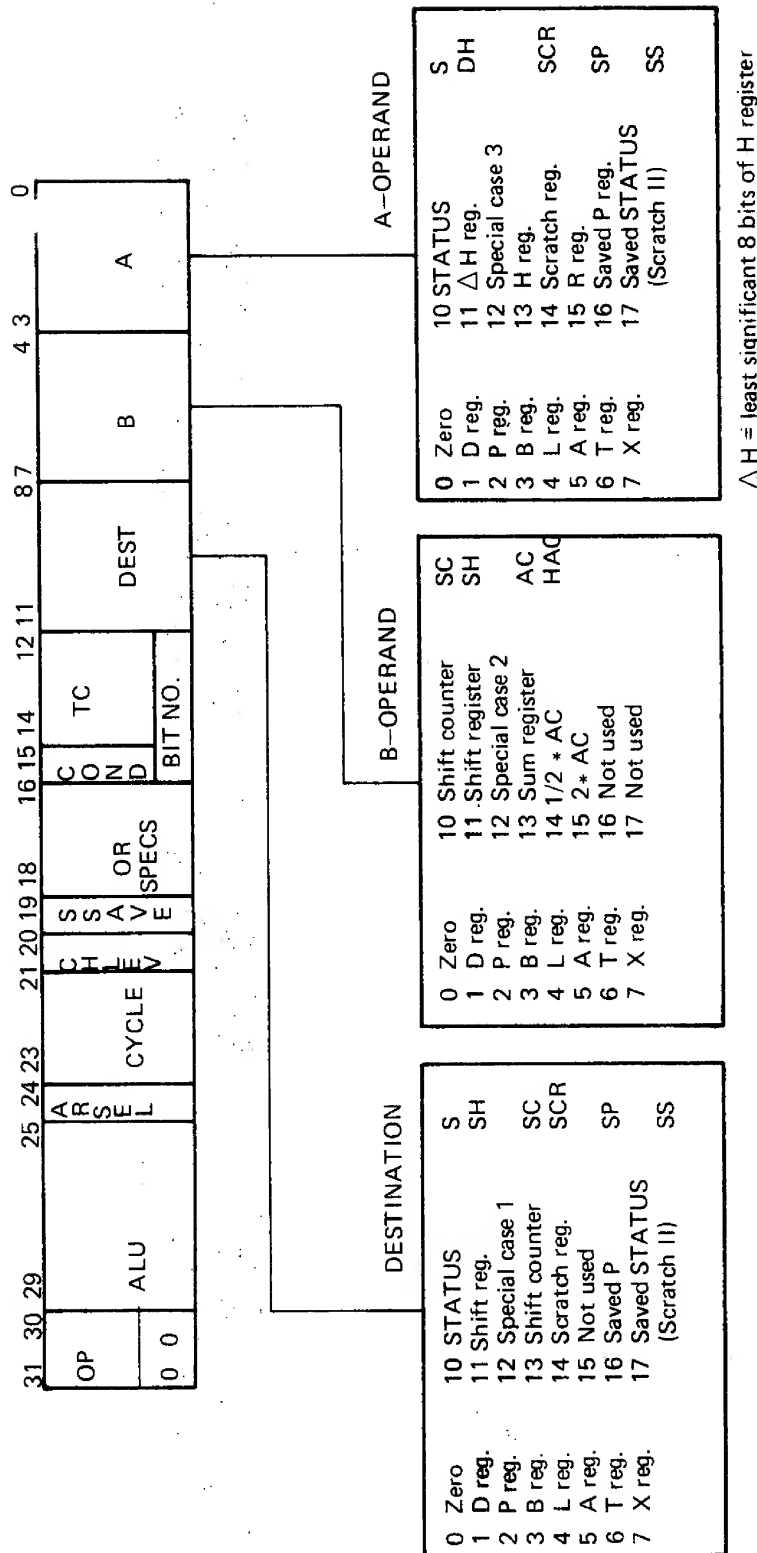
Note: It is the result of a previous arithmetical or logical operation using the most significant unit, which is tested.

Figure 2.1: Arithmetic μ instruction - Bit Assignment 1

				LOGICAL OPERATION MIR 29 = 1		ARITHMETIC OPERATIONS MIR 29 = 0	
MIR				Function	Mne	Function	Mne
28	27	26	25				
0	0	0	0	\overline{B}	BDIRC	B-1	BM1
0	0	0	1	$\overline{B} \cdot \overline{A}$	ANDC		
0	0	1	0	$\overline{B} + A$	ORCB	B-A-1	BMAM1
0	0	1	1	LOGICAL 1	ONE	B	BD1
0	1	0	0	$\overline{B} + \overline{A}$	ORC		
0	1	0	1	\overline{A}	ADIRC	B + A + carry	PLUS ADDC
0	1	1	0	$B \nabla A$	EXORC	(B-A-1) + carry	BMAM1 ADDC
0	1	1	1	$B + \overline{A}$	ORCA		
1	0	0	0	$\overline{B} \cdot A$	ANCB		
1	0	0	1	$B \nabla A$	EXOR	B + A	PLUS
1	0	1	0	A	ADIR		
1	0	1	1	B + A	OR		
1	1	0	0	LOGICAL 0	ZERO		
1	1	0	1	$B \cdot \overline{A}$	ANDCA	B + A + 1	PLUS ADD1
1	1	1	0	$B \cdot A$	AND	B-A	BMINA
1	1	1	1	B	BDIR	B + 1	BDI ADD1

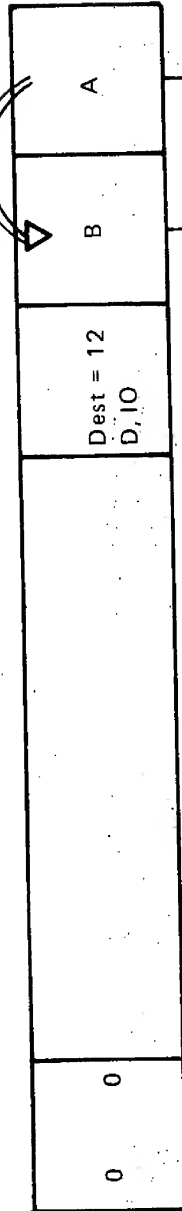
Table 2.1: Function Select Codes

ARITHMETIC:

Figure 2.2: Arithmetic μ -instruction — Bit Assignment II

ARITHMETIC

TRR - Output from Processor



TRR - A Operand field transferred to internal register specified in B operand field

INTERNAL REGISTERS	
TRR	MNE
0 Opr. Panel control reg.	PAC
1 Status	S
2 Lamp Register	LMP
3 Paging Control Register	PCR
4 Miscellaneous Register	MIS
5 Internal Interrupt Enable Register	IEE
6 Priority Interrupt det.	PID
7 Priority Interrupt Enable	PIE
	MNE
10 Not assigned	
11 Not assigned	
12 Not assigned	
13 Computed Address Register	CAR
14 Instruction Register	IR
15 Not assigned	
16 Output to I/O System	BIO

0 Zero	10 STATUS	S
1 D register	11 ΔH register	DH
2 P register	12 Not assigned	H
3 B register	13 H register	SCR
4 L register	14 Scratch register	
5 A register	15 R register	SP
6 T register	16 Saved P register	SS
7 X register	17 Saved STATUS (Scratch II)	

MISC. REG. FORMAT:	
BIT	
0	IOF
1	ION
2	POF
3	PON
4	MCALL MUST BE SET/RESET
5	MOPC

NOTE: ALL numbers are in octal

Figure 2.3: Special Case 1

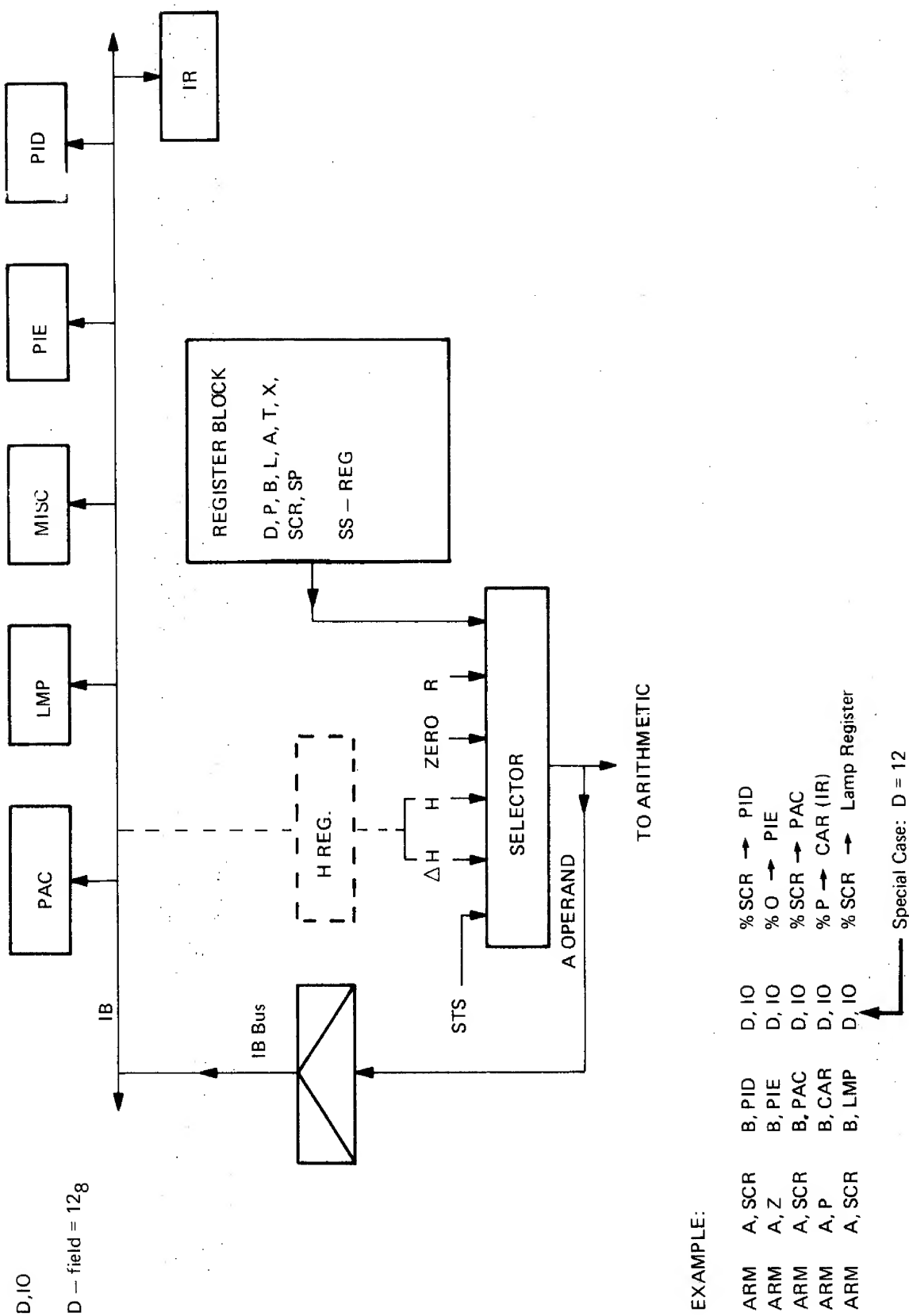
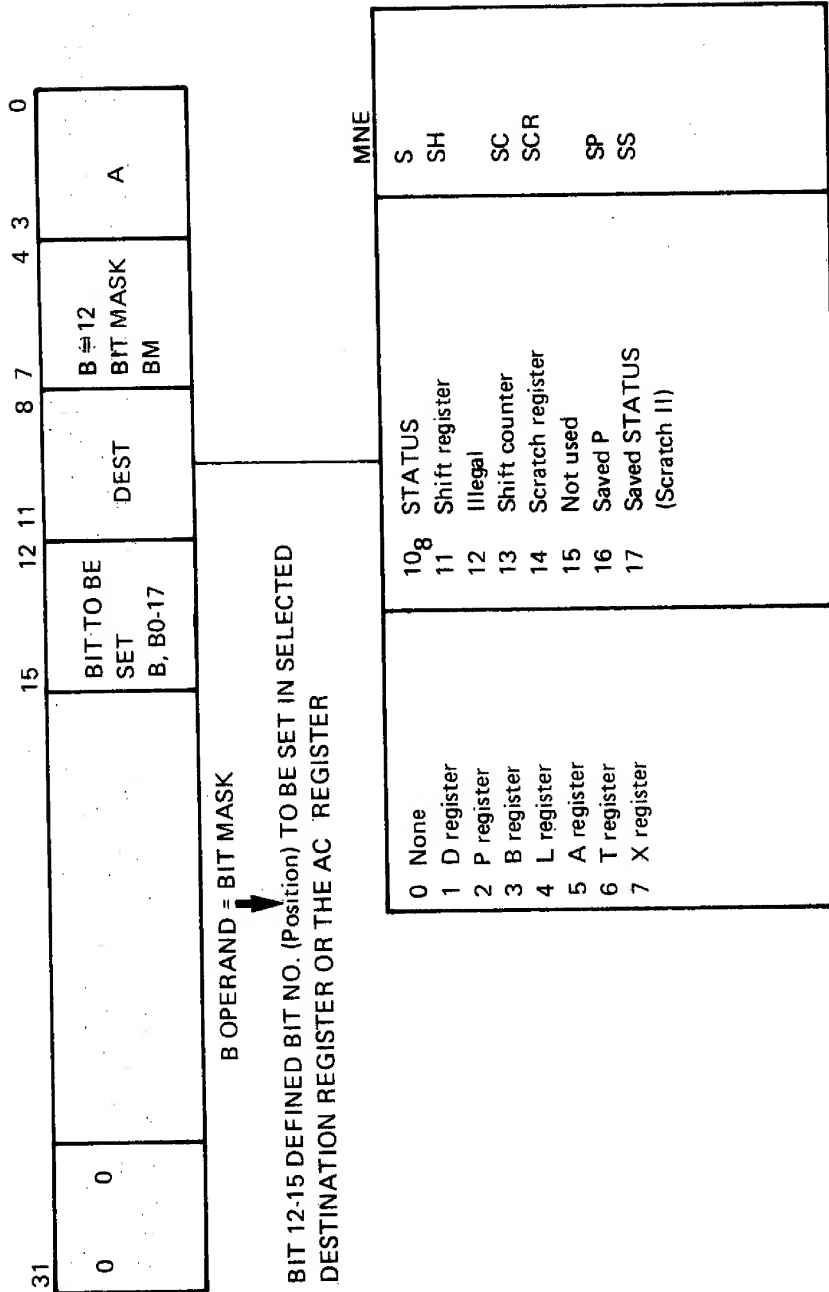


Figure 2.4: Special Case 1 — Illustration

ARITHMETIC

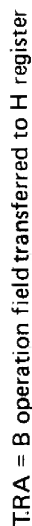


EXAMPLE:

LOGM BDIR B, B4 D, SCR
 ARM CR BM1 B, B5 D, SS

% 20 (BIT 4 = 1) → SCR
 % 40 (BIT 5 = 1) → SS : 37 → SS

Figure 2.5: Special Case 2



All numbers are in octal. On Memory Control the TRXX signals are in decimal.

EXAMPLE:

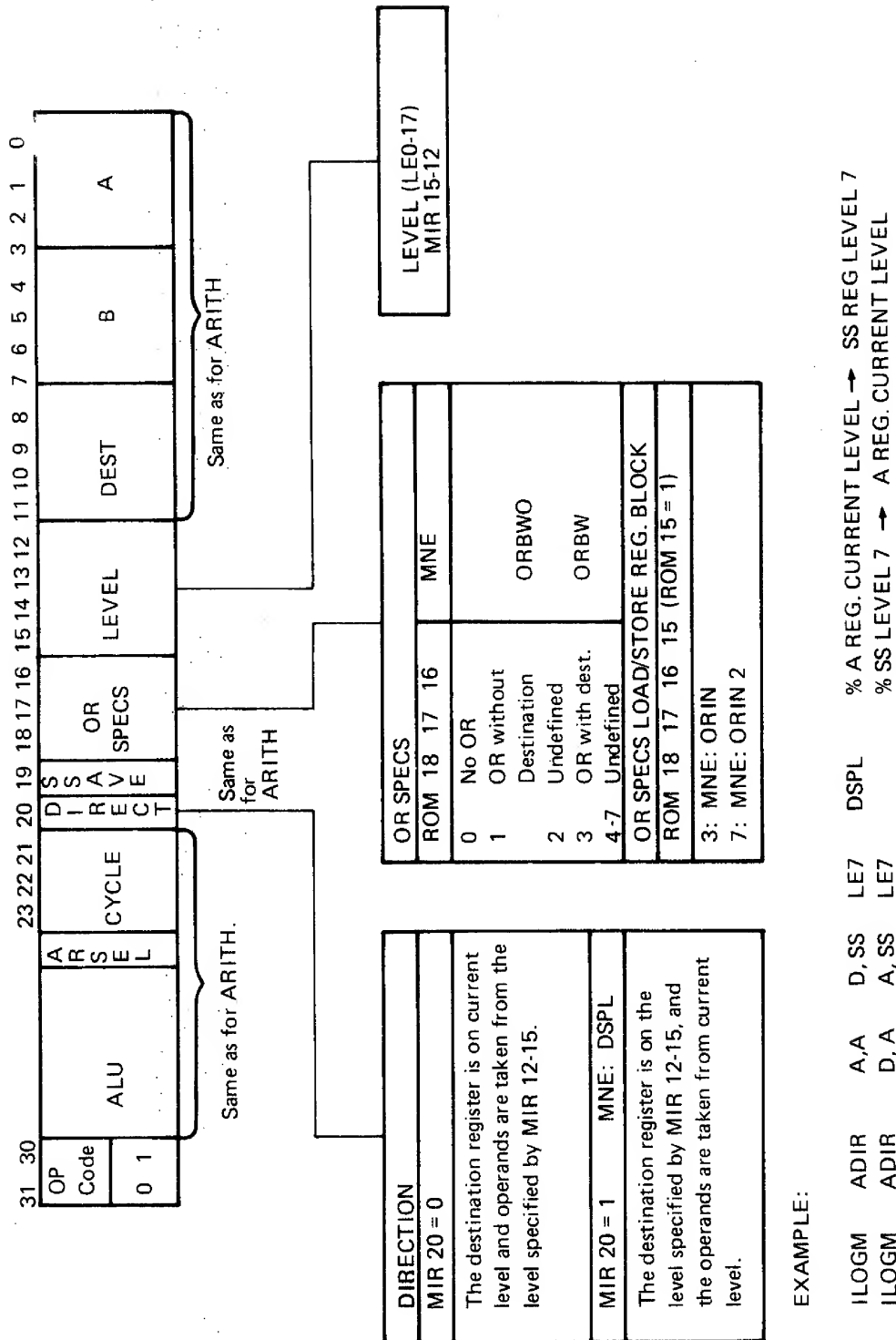
LOGM	A, IO	BIO	% I/O BUS \rightarrow H REG.
LOGM	A, IO	B, PES	% PES \rightarrow H REG.
ARM	A, IO	B, PIM	% PIM (Decoded PIM) \rightarrow H
ARM	A, IO	B, PAS	% PANEL STATUS \rightarrow H

Figure 2.6: Special Case 3

2.2 *THE INTERBLOCK MICROINSTRUCTION*

Refer to Figure 2.7. The INTERBLOCK microinstruction is used for interlevel communication, i.e., for implementing the IRR and IRW instruction. The Interblock instruction is also used by the microprogram for saving and returning of information from scratch registers on different levels. Bit 20 is used for defining the direction of communication as described in Figure 2.7. The two levels to communicate between is always the current level, as specified by PIL — Current Program Level indicator, and the level specified by bits 12 to 15.

INTERBLOCK

Figure 2.7: Interblock μ -instruction - Bit Assignment

2.3 THE JUMP MICROINSTRUCTION

Refer to Figure 3.9. The JUMP instruction may be divided into:

Unconditional JUMP	— JMP
Conditional JUMP	— CJMP
Privileged Instruction JUMP	— JMP PRIV
Computed Address Register JUMP	— JMP, CAR

The JMP instruction takes bits 0 - 11 as an *absolute* address.

The CJMP takes bits 0 - 11 as an absolute jump address if the specified condition is TRUE. If the specified condition is FALSE, the instruction following the CJMP will be executed.

The JMP PRIV instruction is used to generate Privileged Instruction Internal Interrupt (bit 6 in IIC if the privileged instruction executed is on Ring 0 or Ring 1). IOX, IOT and IDENT are decoded separately on 1058 Interrupt Control.

The JMP, CAR instruction is used during subroutine handling and is analogue to the EXIT machine instruction in that the absolute jump address is taken from Computed Address Register — CAR, which contains the main program return address.

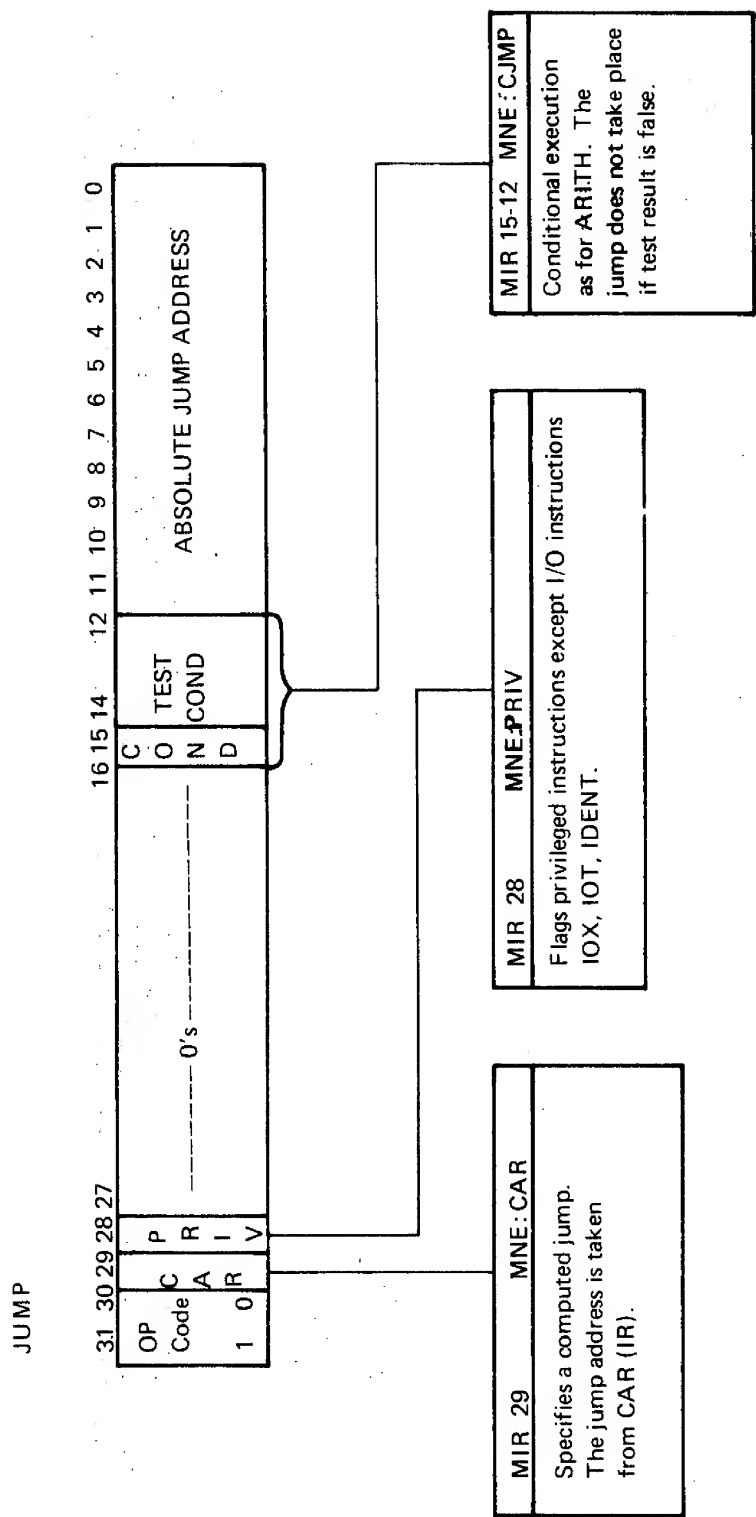


Figure 2.8: Jump μ -instruction — Bit Assignment

2.4 THE LOOP MICROINSTRUCTION

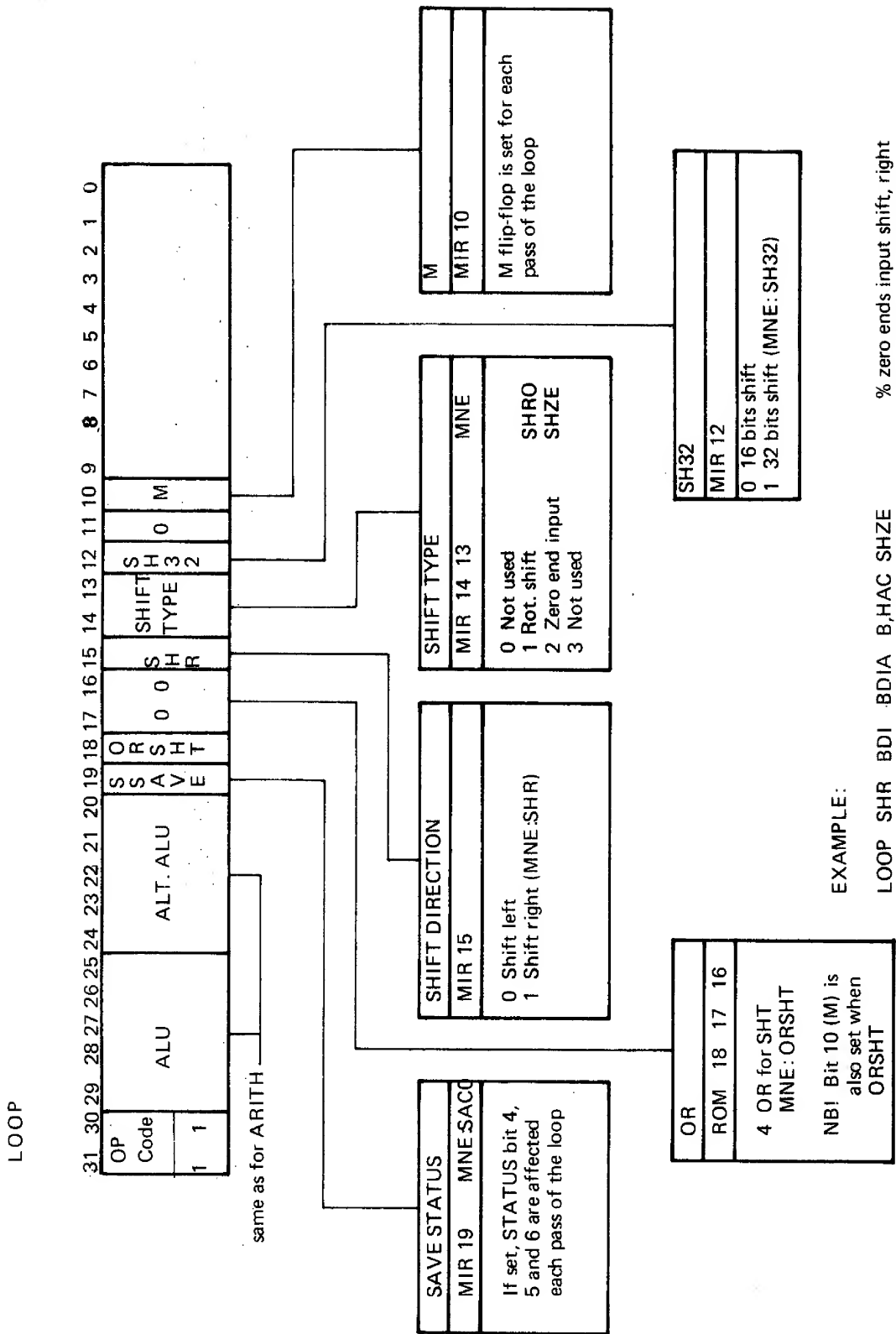
Refer to Figure 2.10. The LOOP instruction is used in all SHIFT, MULTIPLY and DIVIDE operations. When execution of a LOOP instruction is started, the instruction will repeatedly be executed until a specified terminating condition occurs. In other words, the LOOP instruction will remain in the Microinstruction Register and no incrementing of the MPC (Microprogram Counter) will take place before the terminating condition is met.

During shift operations, for instance, the LOOP instruction will be executed as many times as the number of shifts specified.

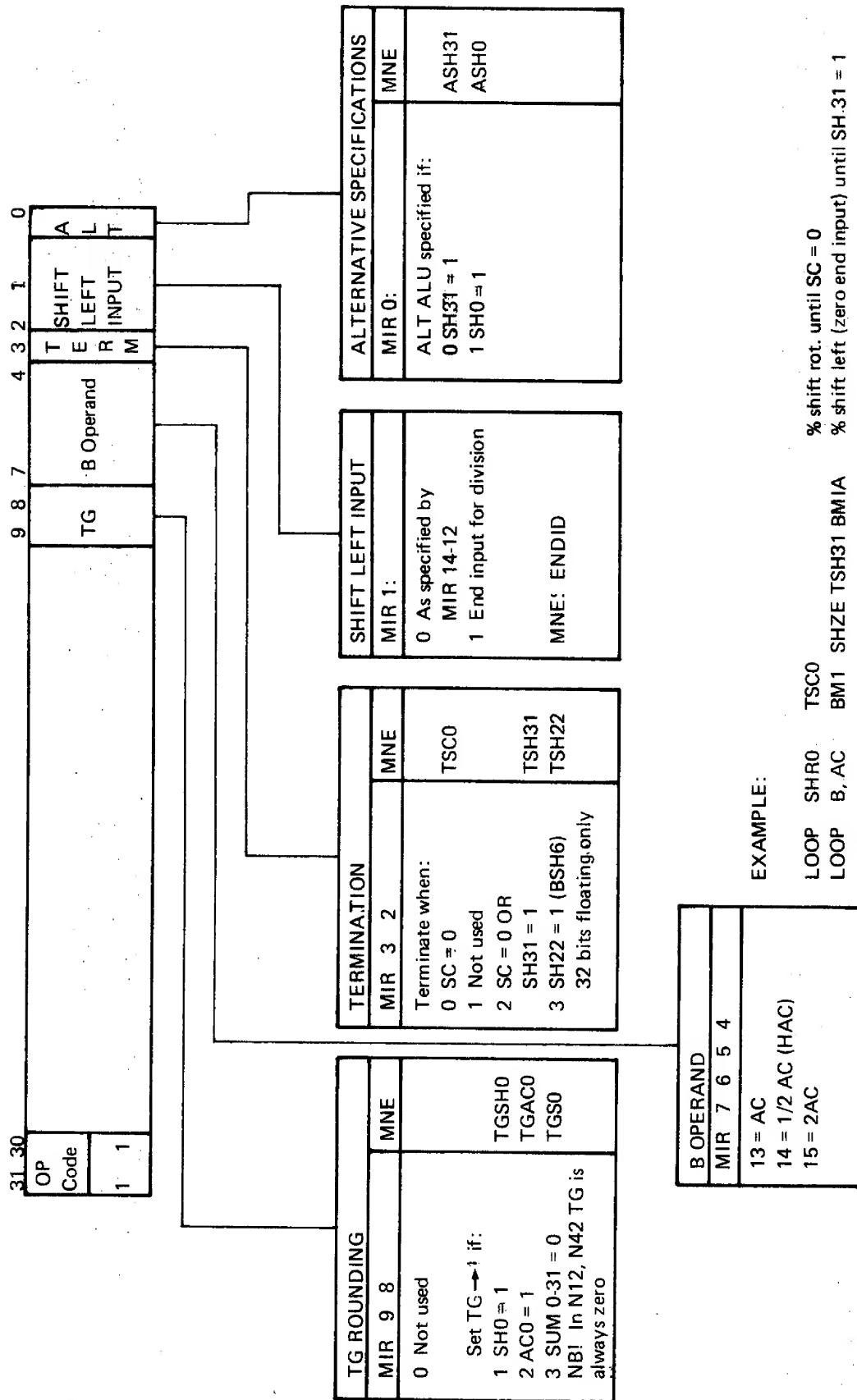
The LOOP instruction may specify any arithmetical or logical operation as for the ARITHMETIC instruction. However, the LOOP instruction may specify operations on both ALU's in the same instruction (depending on bit 0). The LOOP instruction may, thus, effectively operate a 32 bits ALU. This is the case for all floating and double precision instructions.

Refer to Figure 2.10. Bit 0 controls the Alternative Arithmetic function select. If bit 0 = 0, the alternative function select will be used by both ALU's if the most significant arithmetic module's shift register bit 15 (SH_{31}) = 1. This is used by the multiply routines.

When bit 0 = 1, the alternative function select will be used if least significant arithmetic module's shift register bit 0 (SH_0) = 1. This is used by the divide routines.

Figure 2.9: Loop μ -instruction -- Bit Assignment I

LOOP

Figure 2.10: Loop μ -instruction — Bit Assignment II

3 THE MICROPROGRAM

3.1 MICMAC – MICRO MAC MNEMONIC TABLE

A,A	A register as A-operand
A,B	B register as A-operand
A,D	D register as A-operand
AC	Temporary Sum or Accumulator Register
ADD1	Forced Carry Input
ADDC	Add Carry Input
A,DH	Lower 8 bits of H with sign extension as A operand
ADIR	A-operand Direct through Arithmetic = A
ADIRC	A-operand Direct Complemented = \bar{A}
A,H	H register as A-operand
A,IO	Special Case: Internal Register specified as B-operand to H register
A,L	L register as A-operand
ALD	Automatic Load Descriptor
AND	Logical AND = $A \cdot B$
ANDC	AND complement – NAND = $\overline{A \cdot B}$
ANDCA	$\bar{A} \cdot B$
ANDCB	$A \cdot \bar{B}$
A,P	CP register as A-operand
A,R	R register as A-operand
ARL	Arithmetic operation least significant unit
ARM	Arithmetic operation most significant unit

A,S	STATUS as A-operand
A,SCR	SCRATCH register as A-operand
ASH0	Alternative ALU specs. if SH0 = 1
ASH31	Alternative ALU specs. if SH31 = 1
A,SP	SAVED P as A-operand
A,SS	SAVED STATUS as A-operand
A,T	T register as A-operand
A,X	X register as A-operand
A,Z	Zero as A-operand
B,A	A register as B-operand
B,AC	AC register (Temporary SUM or ACCUMULATOR) as B-operand
B,2AC	2 · AC as B-operand
B,ALD	ALD — Automatic Load Descriptor as B-operand
B,B	B register as B-operand
B,B0-17	Bit number is one in Bit Mask as B-operand
B,CAR	Computed Address Register as B-operand
B,D	D register as B-operand
BDI	B-operand direct during arithmetic operation
BDIA	B-operation direct alternative ALU Specs.
BDIR	B-operand direct during logical operation
B,HAC	1/2 AC as B-operand
BIO	I/O bus as source when A,IO: I/O bus as dest. when D,IO
B,IR	IR as B-operand
BIR	IR as B-operand
BIR3	IR0-3 as B-operand

B,IR3	IR0-3 as B-operand
B,L	L register as B-operand
B,LMP	Lamp register as B operand
BM	B-operand = BIT MASK
BM1	B-operand minus 1
BM1A	B-operand minus 1, alternative ALU specs.
BMAA	B-operand — A-operand alternative ALU spec.
BMAM1	B-operand — A-operand — 1 ($B-A-1$)
BMINA	B-operand — A-operand ($B-A$)
B,MIS	Miscellaneous register as B-operand
BMISC	Miscellaneous register as B-operand
B,MPC	Micro Program Counter as B-operand
B,OPR	Panel Switch Register as B operand
B,P	CP register as B-operand
B,PAC	Panel Control as B-operand
B,PAS	Panel Status as B-operand
B,PES	Memory Error Status register as B-operand
B,PID	Priority Interrupt Detect as B-operand
B,PIE	PIE as B-operand
B,PIM	Decoded PIL as B-operand
B,SC	Shift Counter as B-operand
B,SH	Shift register as B-operand
B,T	T register as B-operand
B,X	X-register as B-operand
B,Z	Zero as B-operand

CALL	Jump to subroutine
CAR	Jump address from CAR (Computed Address Register)
CARL	Conditional Arithmetic least significant unit
CARM	Conditional Arithmetic most significant unit
CEATR	Cycle 1, Effective address to R -- no request
CFC	Cycle 3, Fetch
CHLEV	Change program level
CJMP	Conditional jump
CLOGM	Conditional logical operation most significant unit
CO17	Conditional bit set and condition 7
COND	Condition bit set bit 15 in ROM
CPTR	Cycle 2, Current P register to R register
CR	Cycle 7, Read contents of effective address
CRR1	Cycle 6, Read contents of effective address + 1
CW	Cycle 5, Write into effective location
CWR1	Cycle 4, Write into effective location + 1
DH	Lower 8 bits of H sign extended
DNO	Device number
D,A	A register as destination register
D,B	B-register as destination
D,D	D register as destination
D,IO	Special case, A-operand transferred to internal registers
D,L	L register as destination
D,P	CP register as destination

D,S	Status register as destination
D,SS	Saved Status as destination
D,SCR	Scratch register as destination
D,SH	Shift register as destination
D,SP	Saved P as destination
DSPL	Destination register on level specified by bits 12-15 in ROM
D,T	T register as destination
D,X	X register as destination
ENDID	End input for division
EXOR	Exclusive OR
EXORC	Exclusive OR complement
GREM	Greater Magnitude
HAC	$1/2 \cdot AC$
IARM	Interblock arithmetic operation most significant unit
IR	Instruction Register
IR3	Instruction Register bit 0-3. Register number in IR0-3 as destination register
ILOGM	Interblock logical operation most significant unit
JMP	Jump
LE0-17	Level number specified
LMP	Lamp register
LOGL	Logical operation least significant unit
LOGM	Logical operation most significant unit
LOOP	Loop instruction

MIS	Miscellaneous register
MPC	Micro Program Counter
NEG	Test for negative
NEGM	Test for negative magnitude
NZERO	Test for not zero
OR	Inclusive OR
ORBW	OR for bit operations with destination
ORBWO	OR for bit operation without destination
ORC	OR complement ($\overline{A + B}$)
ORCA	$\overline{A} + B$
ORCAR	OR with CAR
ORCB	$A + \overline{B}$
ORIN	OR for Interblock
ORIN2	OR for Interblock
ORROP	OR for Register Operations
ORSHT	OR for Shift
ORSKP	OR for SKP
ORSW2	OR for SWAP cycle 2
ORSW3	OR for SWAP cycle 3
PAC	Panel Control Register
PAS	Panel Status Register
PCR	Paging Control Register
PES	Memory Error Status Register
PIM	Decoded PIL
PLUS	A-operand + B-operand

PLUSA	A-operand + B-operand alternative specs.
POS	Test for positive
POSM	Test for positive magnitude
PRIV	Privileged instructions
S	STATUS — Register
SACO	Save Carry and Overflow
SC	Shift Counter
SCR	Scratch Register
SH	Shift Register
SH32	32 bits shift
SHAR	Arithmetic Shift
SHLI	Link end input
SHR	Shift right
SHRO	Rotational Shift
SHZE	Zero end input
SNEG	Sign negative
SPOS	Sign positive
SS	Saved Status Register
TGAC0	TG = 1 if AC0 = 1
TGS0	TG = 1 if SUM0-31 = 0
TGSH0	TG = 1 if SH0=1
TSC0	Terminate when Shift Counter = 0
TSH31	Terminate when Shift Counter = 0 or SH ₃₁ = 1
TSH22	Terminate when SH22 = 1 (32 bits floating)
Z	Test for zero

3.2

NORD-10 INSTRUCTIONS AND THEIR CORRESPONDING ENTRY-POINTS

AAA	: 352	IRW	: 256	RCLR	: 230
AAB	: 350	JAF	: 306	RDCR	: 231
AAT	: 354	JAN	: 302	RDIV	: 207
AAX	: 356	JAP	: 300	REXO	: 224 - 225
AAD	: 130	JAZ	: 304	RINC	: 232
AND	: 134	JMP	: 152	RMPY	: 205
BANC	: 374	JNC	: 312	RORA	: 226 - 227
BAND	: 375	JPC	: 310	RSUB	: 233
BLDA	: 373	JPL	: 156	SAA	: 342
BLDC	: 372	JXN	: 316	SAB	: 340
BORA	: 377	JXZ	: 314	SAD	: 274
BORC	: 376	LBYT	: 211	SAT	: 344
BSET	: 360 - 363	LDA	: 122	SAX	: 346
BSKP	: 364 - 367	LDD	: 112	SBYT	: 213
BSTA	: 371	LDF	: 116	SHA	: 270
BSTC	: 370	LDT	: 124	SHD	: 264
COPY	: 230	LDX	: 126	SHT	: 260
DNZ	: 250	LRB	: 252	SKP	: 200
EXIT	: 230	MCL	: 240	SRB	: 252
EXR	: 203	MIN	: 120	STA	: 102
FAD	: 140	MIX3	: 215	STD	: 110
FDV	: 146	MON	: 254	STF	: 114
FMU	: 144	MPY	: 150	STT	: 104
FSB	: 142	MST	: 240	STX	: 106
IDENT	: 217	NLZ	: 246	STZ	: 100
IOF	: 242	ORA	: 136	SUB	: 132
ION	: 242	POF	: 242	SWAP	: 220 - 221
IOT	: 170	PON	: 242	TRA	: 240
IOX	: 172	RADD	: 230 - 237	TRR	: 240
IRR	: 256	RAND	: 222 - 223	WAIT	: 244

3.2.1 *Special Entry Points*

Entry Point:
(ADR)

0	Entry point for STOP mode. It is automatically entered if the STOP signal is on during a fetch cycle (pushing the STOP button or executing a WAIT instruction).
1	Entry point for MASTER CLEAR. (Pushing the master clear button or power turn-on.)
400	Entry point for program interrupt, both internal and external.
1000	Entry point for operator's panel interrupt. It is entered with 3 milli-seconds interval when a general register or memory is displayed on the operator's panel.
1400	Entry point for coincident operator's panel and program interrupt.
1657	μ program memory check.

3.3

LABELS REFERENCED IN NORD-10 MICROPROGRAM

ACT	: 1756	IEXA	: 1425	OUTCH	: 1735
ACT1	: 1766	IEXA1	: 1424	PANINC	: 1226
ADDF	: 472	IEXAM	: 1421	PANT1	: 1256
ASS8	: 1176	INCH	: 1716	PANT2	: 1257
BANCC	: 1026	INV	: 553	PANTT	: 1236
BANDC	: 1023	IOTC	: 402	POSDV	: 650
BANK	: 1373	IOXR	: 1632	PRLF	: 1430
BIN	: 1645	IRD	: 1365	PUTGC	: 657
BINL	: 1532	KONE2	: 154	QUM	: 1134
BLDAC	: 1020	KONE3	: 1037	RDEP	: 1476
BLDCC	: 1015	LDBC	: 64	RDIVC	: 677
BONE1	: 1022	LDDC	: 336	REAC	: 1077
BORAC	: 1031	LDFC	: 335	REDEP	: 1403
BORCC	: 1034	LEFT	: 74	REGDP	: 1412
BSBAC	: 1002	LEFTB	: 437	RESTA	: 1304
BSBSH	: 361	LOAD	: 1503	RETPA	: 1462
BSKC	: 420	MAS1	: 1602	RETU	: 1224
BSKCC	: 414	MAS2	: 1622	RETU1	: 1223
BSOC	: 416	MASS	: 1601	RETU5	: 1434
BSTAC	: 1011	MCLS	: 331	REX	: 1443
BSTCC	: 1005	MCLS	: 1774	REXAM	: 1321
BSZC	: 422	MCRY1	: 616	RLOOP	: 1201
CHCR	: 1343	MEXM	: 1456	RPANT	: 1717
CIIP	: 1041	MINC	: 41	RPDEP	: 1404
CLC	: 1441	MLOOP	: 1767	RSTRT	: 1576
CRY1	: 501	MM0	: 1661	SADC	: 175
DE0	: 1516	MM1	: 1662	SEEK	: 1535
DEPP	: 1467	MM2	: 1663	SETAD	: 1446
DNZC	: 2	MM3	: 1664	SIKI	: 1536
DOLET	: 1505	MM4	: 1674	SLRB	: 727
DOLL	: 1501	MM00	: 1660	SRB	: 16
EASS8	: 1220	MM41	: 1677	STBC	: 424
EQUAL	: 536	MONC	: 654	STDC	: 166
ERDP	: 1326	MOPC	: 1054	STFC	: 165
ERR	: 1712	MOPCM	: 1043	STFP	: 1307
ETSGN	: 1504	MOPCR	: 1060	STLP	: 1554
EXAM	: 1273	MPYC	: 753	STORB	: 444
EXECC	: 160	MPYDC	: 661	STPR	: 1310
EXRO	: 1300	NDEP	: 1360	STSP	: 1305
EXTN	: 1574	NECHP	: 1160	SUBF	: 525
FADC	: 454	NED1	: 505	SUBF2	: 524
FAFSC	: 456	NLZC	: 52	SUBF3	: 543
FDVC	: 621	NOINV	: 556	SWPC	: 47
FDVO	: 566	NORMA2	: 521	SWPCC	: 46
FETC5	: 341	NRDP	: 1406	TGTN	: 645
FETCH	: 101	NRDP1	: 1410	TRRS	: 327
FETCZ	: 343	NYFAF	: 1075	TTGN	: 613
FMUC	: 572	OUT1	: 1736	TTMMC	: 320
FSBC	: 451	OUT2	: 1743	WAITC	: 772
GETR	: 353	OUT3	: 1754	ZIR6	: 410
IEX	: 1436	OUT8	: 1146	ZTAD	: 567

3.4 *THE MICROPROGRAM LISTING*

0000	JMP 1402	%STOP
0001	JMP 1401	%ASTER CLEAR
0002	%ROUTINE TO CONVERT FROM FLOATING NUMBER IN T, A, D-REG,	
0002	%TO INTEGER NUMBER IN A-REG	
0002	%D-REG AND T-REG IS SET TO ZERO	
0002	DNZC,	ADDRESS:
0003	ARM PLUS A, DH B, T D, A	%T + ΔH → A
0004	ARM PLUS D, SS B, B4 A, A	%A + 20 → SS
0005	LOGM AND B, B16 A, SS	%TEST BIT 16 in SS-REG
0006	CJMP ZERO ZTAD	%JMP IF OVERFLOW
0007	LOGM AND D, D B, B16 A, A	%TEST BIT 16 IN A-REG
0010	CJMP NZERO FETCZ	%JMP IF OVERFLOW
0011	LOGM ADIR A, A D, SC	%SET SHIFTCOUNT
0012	LOOP SHR SHZE TSC0	%SHIFT RIGHT TO SC = 0
0013	LOGM BDIR B, SH D, A	%SH → A
0014	LOGM BDIR B, T	%TEST SIGN
0015	CARM SNEG BMINA A, A B, Z D, A	%INVERT IF SIGN NEG
0016	LOGM CFC ADIR A, Z D, T	%0 → T, FETCH
0017	SRB,	%STORE BLOCK, X -1 → CP
0020	LOGM ADIR A, SP D, P	%UNSAVE CP
0021	ILOGM ADIR ORIN2 A, SP	%READ SP SPES, LEV
0022	LOGM CWR1 A, SP	%STORE SP
0023	ILOGM ADIR ORIN2 A, X	%READ X
0024	ARM CWR1 A, SP	%READ T
0025	ILOGM ADIR ORIN2 A, T	%READ A
0026	ARM CWR1 A, SP	%READ D
0027	ILOGM ADIR ORIN2 A, A	%READ L
0030	ARM CWR1 A, D ORIN2	
0031	ARM CWR1 A, SP	
0032	ILOGM ADIR ORIN2 A, L	

0033
0034
0035
0036
0037
0040
0041
0041
0041
0042
0043
0044
0045
0046
0047
0050
0051
0052
0052
0052

0052
0052
0053
0054
0055
0056
0057
0060
0061
0062
0063
0064
0064
0065

0066	ARM CPTR PLUS A, T B, SH D, P				%T + SH → CP, CP → R
0067	LOGM CRR1 ADIR A, SP D, P				%R + 1) → H, SP → CP
0070	LOGM AND A, X B, B0			74	%TEST LEFT RIGHT
0071	CJMP ZERO LEFT				
0072	ARM BM1 B, B10 D, SH				%377 → SH
0073	LOGM CFC AND A, H B, SH D, A				%BYTE → A-REG, FETCH
0074	ARM BM1 B, B3 D, SC				%7 → SC
0075	ARM PLUS A, H B, Z				%H → AC
0076	LOOP SHR BDI BDIA B, HAC SHZE				
0077	LOGM CFC BDIR B, HAC D, A				%BYTE → A-REG, FETCH
0100					
0100					
0100					
0101	ARM CW A, Z				%STZ, ZERO → EFFECTIVE ADDRESS
0102	ARM CFC				%FETCH REQUEST
0103	ARM CW A, A				%STA, A-REG → EA
0104	ARM CFC				%FETCH
0105	ARM CW A, T				%STI, T-REG, → EA
0106	ARM CFC				%FETCH
0107	ARM CW A, X				%STX, X-REG → EA
0110	ARM CFC				%FETCH
0111	ARM CW A, A				%STD, A-REG → EA
0112	JMP STDC		166		
0113	ARM CR				%LDD, (EA) → H
0114	JMP LDDC		336		
0115	ARM CW A, T				%STF, T-REG → EA
0116	JMP STFC		165		%JUMP TO CONTINUE STF
0117	ARM CR				%LDF, READ REQUEST
0120	JMP LDFC		335		
0121	LOGL CR ADIR A, P D, SH				%MIN, CP → SE READ MEMORY
0122	JMP MNC		41		
0123	ARM CR				%LDA, (EA) → H
0124	LOGM CFC ADIR A, H D, A				%H → A, FETCH REQUEST
0125	ARM CR				%LDT, (EA) → H
	LOGM CFC ADIR A, H D, T				%H → T, FETCH REQUEST

0126	ARM CR		%LDX, (EA) → H
0127	LOGM CFC ADIR A, H D, X		%H → X, FETCH REQUEST
0130	ARM CR		%ADD, (EA) → H
0131	ARM CFC PLUS SACO A, H B, A D, A		%H + A → A, FETCH
0132	ARM CR		%SUB, (EA) → H
0133	ARM CFC BMINA SACO A, H B, A D, A		%A - H → A, FETCH
0134	ARM CR		%AND, (EA) → H
0135	LOGM CFC AND A, H B, A D, A		%A · H → A, FETCH
0136	ARM CR		%ORA, (EA) → H
0137	LOGM CFC OR A, H B, A D, A		%A + H → A, FETCH
0140	LOGM ANDCB B, B1 A, S D, S		%STARTADD. FAD, RESET "TG"
0141	JMP FADC	454	%START FSB, RESET "TG"
0142	LOGM AND CB B, B1 A, S D, S		%START FMU, RESET "TG"
0143	JMP FSBC	451	%START FDV, RESET "TG"
0144	LOGM ANDCB B, B1 A, S D, S		%START MPY, SET SHIFTCOUNTER
0145	JMP FMUC	572	%JMP, EA → R
0146	LOGM ANDCB B, B1 A, S D, S		%R → CP, (R) → H, IR
0147	JMP FDVC	621	%0 → SPECIFIED BIT
0150	LOGM BDIR B, B4 D, SC		%1 → K, FETCH
0151	JMP MPYC	753	%JPL, EA → R, CP → L
0152	ARM CEATR		%R → CP, (R) → H, IR
0153	LOGM CFC ADIR A, R D, P		%REG → SP
0154	LOGM AND CB B, BM ORBW		%TEST EXEC. INSTR IN REG
0155	LOGM CFC OR B, B2 A, S D, S		%JMP IF EXECUT INSTR IN REG
0156	LOGM CEATR ADIR A, P D, L		%INSTRUC. → IR
0157	LOGM CFC ADIR A, R D, P		%A → (R + 1)
0160	LOGM ADIR ORSKP D, SP		%D → (R + 1)
0161	LOGM EXOR A, H B, AC D, SS	343	%JMP IOT CONTINUE
0162	LOGM AND CB A, SS B, SH		
0163	CJMP ZERO FETCZ		
0164	ARM A, SP B, IR D, IO		
0165	ARM CWR1 A, A		
0166	ARM CWR1 A, D		
0167	ARM CFC		
0170	JMP PRIV IOTC	402	

0171	FETC1,	LOGM CFC BDIR B, SH D, D	%SH(L) → D, FETCH
0172	IOX,	LOGM ADIR A, A BIO D, IO	%A - REG → IO-BUS
0173		LOGM A, IO BIO	%IO-BUS → H-REG
0174		LOGM CFC ADIR A, H D, A	%H-REG → A-REG FETCH
0175	SADC,	LOOP ORSHT SH32 TSC0	%SHIFT, TERM:SC = 0
0176		LOGM BDIR B, SH D, A	%SH → A
0177		LOGM CFC BDIR B, SH D, D	%SH(L) → D, FETCH
0200		ARM BMINA ORSKP	%SKP
0201		CARM ORSKP PLUS ADD1 CFC A, Z B, P D, P	%TRUE CP + 1 → CP, FETCH
0202	FETC2,	LOGM CFC BDIR B, 2AC D, D	%2AC(L) → D
0203	EXEC,	ARM BM1 B, B6 D, SH	%EXECUTE, GENERATE MASK
0204		JMP EXECC	160
0205	RMPY,	LOGM ADIR ORSKP D, SP	%FIRST OPERAND → SP, REGISTER MULTIPLY
0206		JMP MPYDC	661
0207	RDIV,	LOGM BDIR B, B4 D, SC	%20 → SC, REGISTER DIVIDE
0210		JMP RDIVC	677
0211	LDB,	LOGM ADIR A, P D, SP	%LOAD BYTE, CP → SP
0212		JMP LDBC	64
0213	STB,	LOGM ADIR A, P D, SP	%STORE BYTE, SAVE CP REG
0214		JMP STBC	424
0215	MPX3,	ARL BM1 B, A D, SCR	%INSTR:(A - 1) * 3 → SCRA → AC
0216		ARL CFC PLUS A, SCR B, 2AC D, X	%3 · AC + SCR → SCR
0217	IDENT	JMP IOX	172
0220		JMP SWPC	47
0221		JMP SWPCC	46
0222		LOGM CFC AND ORROP	%SWAP CM1
0223		LOGM CFC ANDCA ORROP	%RAND, D · S → D
0224		LOGM CFC EXOR ORROP	%RAND CM1, D · S → D
0225		LOGM CFC EXORC ORROP	%REXO, DVS → D
0226		LOGM CFC OR ORROP	%REXO CM1, DVS → D
0227		LOGM CFC ORCA ORROP	%RORA, D + S → D
0230		ARM CFC PLUS ORROP SACO	%OR CM1, D + S → D
0231		ARM CFC BMAM1 ORROP SACO	%RADD, COPY, EXIT, RCLR, D + S → D
0232		ARM CFC PLUS ADD1 ORROP SACO	%RADD CM1, D - S - 1 → D, RDCR
0233		ARM CFC BMINA ORROP SACO	%RADD AD1, D + S + 1 → D, RINC
0234		ARM CFC PLUS ADDC ORROP SACE	%RADD AD1 CM1, D - S → D, RSUB
			%RADD ADC, D + S # C → D

0235	ARM CFC BMAM1 ADDC ORROP SACO			%RADD ADC CM1, D - S - 1 + C → D
0236	ARM CFC PLUS ADD1 ORROP SACO			%RADD AD1 ADC D + S + 1 → D
0237	ARM CFC BMINA ORROP SACO			%RADD AD1 ADC CM1, D - S → D
0240	LOGM AND A, H B, B6			%MST, MCL, TRA, TRR, TEST BIT 6
0241	JMP PRIV TTMMC	320		%IOF, ION, PON, POF
0242	JMP PRIV CIIPP	1041		%READ PID
0243	0			
0244	ARM A, I0 B, PID		772	%START NLZ, 0 → D
0245	JMP PRIV WAITC			
0246	LOGM BDIR B, Z D, D		52	%START DNZ, A → SH
0247	JMP NLZC			
0250	LOGM ADIR D, SH A, A		2	%LOAD/STORE REGISTER BLOCK, CP → SP
0251	JMP DNZC			
0252	LOGM ADIR A, P D, SP		727	%MONITOR CALL, 4 → SCR
0253	JMP PRIV SLRB			
0254	LOGM BDIR B, B4 D, SCR		654	%TEST IRR, IRW
0255	JMP MONC			
0256	LOGM AND A, H B, B7		657	%SHT, H(0 - 5) → SC
0257	JMP PRIV PUTGC			%T → SH
0260	LOGM ADIR A, H D, SC			%SHIFT TO SC = 0
0261	LOGM ADIR A, T D, SH			%SH → T, (R) → H, IR
0262	LOOP ORSHT TSC0			%SHD, H(0 - 5) → SC
0263	LOGM CFC BDIR B, SH D, T			%D → SH
0264	LOGM ADIR A, H D, SC			%SHIFT TO SC = 0
0265	LOGM ADIR A, D D, SH			%SH → D
0266	LOOP ORSHT TSC0			%SHA, H(0 - 5) → SC
0267	LOGM CFC BDIR B, SH D, D			%A → SH
0270	LOGM ADIR A, H D, SC			%SHIFT, TERMINATION:SC = 0
0271	LOGM ADIR A, A D, SH			%SH → A, (R) → H, IR
0272	LOOP ORSHT TSC0			%SAD, H(0-5) → SC
0273	LOGM CFC BDIR B, SH D, A			%A → SH
0274	LOGM ADIR A, H D, SC			
0275	LOGM ADIR A, A D, SH			

0276	LOGM ADIR A, D D, SH		%D → SH (LEAST SIGNIFICANT)
0277	JMP SADC	175	
0300	LOGM CEATR ADIR A, A		%JAP, EA → R, TEST A
0301	CLOGM CFC POS ADIR A, R D, P		%(A > 0):R → CP, (R) → H, IR
0302	LOGM CEATR ADIR A, A		%JAN, EA → R TEST A
0303	CLOGM CFC NEG ADIR A, R D, P		%(A < 0):R → CP, (R) → H, IR
0304	LOGM CEATR ADIR A, A		%JAZ, EA → R, TEST A
0305	CLOGM CFC ZERO ADIR A, R D, P		%A = 0:R → CP, (R) → H, IR
0306	LOGM CEATR ADIR A, A		%JAF, EA → R, TEST A
0307	CLOGM CFC NZERO ADIR A, R D, P		%(A ≠ 0):R → CP, (R) → H, IR
0310	ARM CEATR BDI ADD1 B, X D, X		%JPC, X + 1 → X, EA → R
0311	CLOGM CFC SPOS ADIR A, R D, P		%(X > 0):R → CP, (R) → H, IR
0312	ARM CEATR BDI ADD1 B, X D, X		%JNC, X + 1 → X, EA → R
0313	CLOGM CFC SNEG ADIR A, R D, P		%(X < 0):R → CP, (R) → H, IR
0314	LOGM CEATR ADIR A, X		%JXZ, EA → R, TEST X
0315	CLOGM CFC ZERO ADIR A, R D, P		%(X < 0):R → CP
0316	LOGM CEATR ADIR A, X		%JXN, EA → R, TEST X
0317	CLOGM CFC NEG ADIR A, R D, P		%(X < 0):R → CP, (R) → H, IR
0320	CJMP ZERO ZIR6	410	%JMP IF BIT 6 = 0
0321	LOGM AND A, H B, B7		%TEST BIT 7
0322	CJMP ZERO TRRS	327	%JMP IF BIT 7 = 0
0323	ARM A, IO BIR3		%MST, REG → H
0324	LOGM OR A, H B, A D, SP		%H + A → SP
0325	LOGM ADIR A, SP BIR3 D, IO		%SP → REG
0326	JMP FETCH	101	
0327	LOGM A, A BIR3 D, IO ADIR		%TRR, A → REG
0330	JMP FETCH	101	
0331	ARM A, IO BIR3		%MCL, REG → H
0332	LOGM ANDCB A, H B, A D, SP		%N(A), H → SP
0333	LOGM A, SP BIR3 D, IO ADIR		%SP → REG
0334	JMP FETCH	101	
0335	LOGM CRR1 ADIR A, H D, T		%H → T, (R + 1) → H
0336	LOGM CRR1ADIR A, H D, A		%H → A, (R + 1) → H
0337	LOGM CFC ADIR A, H D, D		%H → D, (R) → H, IR

0340	FETC5,	LOGM CFC ADIR A, DH D, B	%H (0-7) → B, FETCH
0341		LOGL BDIR B, AC D, D CFC	%AC (L) → D
0342	FETCZ,	LOGM CFC ADIR A, DH D, A	%SAA, H (0-7) → A, (R) → H, IR
0343		LOGM CFC OR A, S B, B3 D, S	%1 → Z, FETCH
0344		LOGM CFC ADIR A, DH D, T	%SAT, H(0-7) → T, (R) → H, IR
0345		JMP *	
0346		LOGM CFC ADIR A, DH D, X	%SAX, H(0-7) → X, (R) → H, IR
0347	FETC3,	ARM CFC BM1 B, T D, T	%DECR, EXPONENT, FETCH
0350		ARM CFC PLUS SACO A, DH B, B D, B	%AAB, H(0-7) + B → B (R) → H, IR
0351		0	
0352		ARM CFC PLUS SACO A, DH B, A D, A	%AAA, H (0-7) + A → A, (R) → H, IR
0353	GETR,	ILOGM CFC ADIR ORBW D, A	%REG → A, FETCH, IRR
0354		ARM CFC PLUS SACO A, DH B, T D, T	%AAT, H(0-7) + T → T, (R) → H, IR
0355	OVERF,	LOGM OR B, B5 A, S D, S CFC	%SET STATIC OVERF, FETCH
0356		ARM CFC PLUS SACO A, DH B, X D, X	%AAX, H(0-7) + X → X, (R) → H, IR
0357	FETC4,	LOGL CFC BDIR B, 2AC D, D	%2 · AC(L) → D, FETCH
0360		LOGM ANDCB B, BM ORBW CFC	%BSET ZERO
0361	BSBSH,	LOGM OR B, BM ORBW CFC	%BSET ONE
0362		LOGM EXOR B, BM ORBW CFC	%BSET BCM
0363		JMP BSBAC	%JMP TO BSET BAC
0364		JMP BSZC	%JMP TO BSKP ZERO
0365		JMP BSOC	%JMP TO BSKP ONE
0366		JMP BSKCC	%JMP BSKP K0
0367		JMP BSKC	%JMP BSKP K
0370		JMP BSTCC	%JMP BSTC
0371		JMP BSTAC	%JMP BSTA
0372		JMP BLDC	%JMP BLDC
0373		JMP BLDAC	%JMP BLDA
0374		JMP BANCC	%JMP BANC
0375		JMP BANDC	%JMP BAND
0376		JMP BORCC	
0377		JMP BORAC	%JMP BORA
0400		LOGM CHLEV ADIR A, P D, SP	%INTERRUPT, SAVE CP-REG
0401		LOGM CFC ADIR A, SP D, P	%SAVE CP AND FETCH ON NEW LEVEL
0402	IOTC,	LOGM ADIR A, A BIO D, IO	%A-REG → IO-BUS

0403	LOGM A, IO BIO				~IO-BUS → H-REG
0404	LOGM ADIR A, H D, A				%H → A-REG
0405	LOGM A, IO B, PES				%PES → H-REG
0406	LOGM ADIR A, H				%TEST BIT 15 PES (N-1 SKIP)
0407	CARM CFC SNEG BEI ADD1 B, P D, P				%IF MPC 15 = 1 THEN P + 1 → P
0410	LOGM AND A, H B, B7				%TEST BIT 7
0411	CJMP NZERO MCIS			1774	%JMP IF BIT 7 = 1
0412	ARM A, IO BIR 3				%TRA, REG → H
0413	LOGM CFC ADIR A, H D, A				%H → A, (R) → H, IR
0414	LOGM AND A, S B, B2				%TEST K
0415	CJMP NZERO BSZC			422	%JMP IF K = 1
0416	LOGM AND ORBWO B, BM				%TEST BIT
0417	CARM CFC NZERO BDI ADD1 B, P D, P				%IF SPECIFIED BIT = 1:CP + 1 → CP, FETCH
0420	LOGM AND A, S B, B2				%TEST K
0421	CJMP NZERO BSOC			416	%JMP IF K = 1
0422	LOGM AND ORBWO B, BM				%TEST BIT
0423	CARM CFC ZERO BDI ADD1 B, P D, P				%IF SPECIFIED BIT = 0:CP + 1 → CP
0424					
0424	STBC,				
0424	ARM BDI B, X				%X → AC
0425	ARM BM1 B, HAC D, SH				%1/2*AC - 1 → SH
0426	ARM CPTR PLUS A, T B, SH D, P				%T + SH → CP, CP → R
0427	ARM PLUS A, SP B, Z D, P CRR1				%(R + 1) → H
0430	ARM CPTR PLUS A, T B, SH D, P				
0431	ARM BM1 B, B10 D, SCR				%377 → SH
0432	LOGM AND B, B0 A, X				%TEST LEFT RIGHT
0433	CJMP ZERO LEFTB			437	
0434	LOGM AND A, SCR B, A D, SS				%MASK OUT RIGHT PART
0435	LOGM ADIRC A, SCR D, SH				
0436	JMP STORB			444	
0437	ARM BM1 B, B3 D, SC				%7 → SC
0440	LOGL BDIR B, A				%SHIFT TO LEFT BYTE
0441	LOOP BDI BDIA B, 2AC				
0442	LOGL BDIR B, 2AC D, SS				
0443	LOGM ADIR A, SCR D, SH				

0444	STORB,	LOGM AND A, H B, SH D, SH	%SP → CP
0445		LOGM ADIR A, SP D, P	%NEW BYTE + OLD BYTE → SS-REG
0446		LOGM OR A, SS B, SH D, SS	%STORE BYTE
0447		LOGM CWR1 A, SS	%FETCH
0450		LOGM CFC	
0451	FSBC,	ARM CR NM1 B, B5 D, SS	%37 → SS
0451		LOGM EXOR B, B17 A, H D, SCR	%EXPONENT → SCR, INVERT BIT 17
0452		JMP FAFSC	
0453	FADC,	ARM CR BM1 B, B5 D, SS	%37 → SS
0454		LOGM ADIR A, H D, SCR	%H → SCR
0455		LOGM ANDCB B, B17 A, T D, SH	%RESET SIGN BIT
0456	FAFSC,	LOGM ANDCB B, B17 A, SCR D, SP	%RESET SIGN BIT
0457		ARM CRR1 BM1A B, SH A, SP D, SC	%SP - SH → SC
0460		CJMP SNEG NED1	%JMP IF NUMB. IN MEMORY GREATEST
0461		CJMP ZERO EQUAL	%JMP EXPONENTS EQUAL
0462		LOGM ANDCA B, AC A, SS	%TEST AC < 40
0463		CJMP NZERO FETCH	%NO MANTISS OVERLAP IF NOT ZERO
0464		LOGM CRR1 ADIR A, H D, SH	%H → SH
0465		LOGL ADIR A, H D, SH	%H → SH(L)
0466		LOOP SHR SH32 TGS0 SHZE	%SHIFT RIGHT TERM:SC = 0
0467		LOGM EXOR A, SCR B, T	%TEST EQUAL SIGNS
0470		CJMP SNEG SUBF	%JMP IF DIFFERENT SIGN
0471		LOGL BDIR B, SH D, SCR	%SH(L) → SCR
0472	ADDF,	ARM PLUS A, SCR B, D D, D SACO	%ADD. OF LEAST MANT.
0473		ARM PLUS ADDC A, A B, SH D, A SACO	%ADD. OF MOST. MANT.
0474		CJMP GREM CRY1	%JMP IF CRY
0475		LOGM* AND B B1 A, S	%TEST TG * CFC is added in N12/42
0476	NOCRY,	CJMP ZERO FETCH	%ZERO:FINISHED
0477		LOGM CFC OR A, D B, B0 D, D	
0500			
0501		LOGL A, D ADIR	%D → AC(L)
0501	CRY1,	LOGL BDIR B, HAC D, D	%1/2 AC(L) → D
0502		LOGM BDIR B, HAC D, A	%1/2 AC → A CRY → AA
0503			

0504	ARM PLUS B, B0 A, T D, T CFC		%T + 1 → T, FETCH
0505			
0505	NED1, LOGM ORC A, SS B, AC		%TEST AC > -40
0506	CJMP NZERO NORMA2	521	%NO MANTISSE OVERLAP IF NOT ZERO
0507	LOGL ADIR A, D D, SH		
0510	LOGM ADIR A, A D, SH		
0511	LOGM CRR1 ADIR A, H D, SP		%READ MOST SIGNIF MANT
0512	LOGM ADIR A, H D, D		
0513	LOGM ADIR A, SP D, A		
0514	LOOP SHR SH32 TGSH0 SHZE TSC0		%SHIFT RIGHT, TERMIN; SC = 0
0515	LOGM EXOR A, SCR B, T		%TEST FOR EQUAL SIGN
0516	CJMP SNEG SUBF2	524	%JMP TO SUBTRACTION IF UNEQUAL
0517	LOGM ADIR A, SCR D, T		
0520	JMP ADDF	472	
0521			
0521	NORMA2, LOGM CRR1 ADIR A, H D, A		%H → A READ LEAST SIGNIF MANT.
0522	LOGM ADIR A, H D, D		
0523	LOGM CFC ADIR A, SCR D, T		%EXPONENT → T-REG, FETCH
0524			
0524			
0524	SUBF2, LOGM ADIR A, SCR D, T		%LEAST NUMB → REG. ON A-BUS
0525	LOGL BDIR B, SH D, SCR		%D - SCR → SCR
0526	LOGM BDIR B, SH D, SP		%A - SP → SH
0527	ARM BMINA B, D A, SCR D, D S ACO		%TEST TG
0530	ARM BMAM1 ADDC B, A A, SP D, SH		
0531	LOGM AND B, B1 A, S		%SET D0 → 1
0532	CJMP ZERO *2		
0533	LOGL OR B, B0 A, D D, D		
0534	LOGL ADIR A, D D, SH		
0535	JMP NOINV +1	557	
0536			
0536			
0536	EQUAL, LOGM EXOR A, SCR B, T		%TEST EQUAL SIGNS
0537	CJMP SNEG SUBF3		%UNEQUAL JMP TO SUB
0540	LOGM ADIR A, H D, SH CRR1		%H → SH, READ LEASTS. MANT

0541	LOGM ADIR A, H D, SCR		%LEAST SIGN. MANT → SCR
0542	JMP ADDF +1	473	
0543			%H → SCR READ LEASTS. MANT
0543	LOGM ADIRC A, H D, SCR		%H → SP, ONE'S COMPL
0544	LOGM ADIRC A, H D, SCR		%D + SCR → SCR
0545	ARM PLUS A, SCR B, D D, SCR SACO		%A + SP → SP
0546	ARM PLUS ADDC A, SP B, A D, SH SACO		%INVERT IF CARRY = 0
0547	CJMP NEGM INV	553	%SCR + 1 → SCR
0550	ARM PLUS ADD1 A, SCR D, SCR SACO		%SH + CARRY → SH
0551	ARM PLUS ADDC B, SH D, SH		
0552	JMP NOINV	556	
0553	LOGL EXOR B, B17 A, T D, T		%INVERT SIGN
0554	LOGL ADIRC A, SCR D, SCR		%ONE'S COMPL. OF SCR → SCR
0555	LOGM BDIR B, SH D, SH		%ONE'S COMP. OF SH → SH
0556	LOGL ADIR A, SCR D, SH		%SCR → SH
0557	LOGM BDIR B, B5 D, SC		%40 → SC
0560	LOGL BDIR B, T		%T → AC(L)
0561	LOOP BM1 BM1A TSH31 SH32 B, AC		%SHIFT LEFT TO SH31 = CRY
0562	LOGL BDIR B, AC D, T		%EXPONENT → T-REG
0563	LOGM BDIR B, SH D, A		%SH → A
0564	CJMP SPOS ZTAD	567	%RESULT ZERO
0565	LOGL CFC BDIR B, SH D, D		%SH(L) → D, FETCH
0566			%1 → Z
0566	LOGM OR B, B3 A, S D, S		%0 → T
0567	LOGM ADIR A, Z D, T		%0 → A
0570	LOGM ADIR A, Z D, A		%0 → D, FETCH
0571	LOGM CFC ADIR A, Z D, D		
0572			%40 → SC, READ EXPO.
0572	LOGM CR BDIR B, B5 D, SC		%H + T → SP
0573	ARM PLUS A, H B, T D, SP		%RESET BIT 17 IN SP
0574	LOGM ANDCB B, B17 A, SP D, SP		%TEST SIGN
0575	LOGM CRR1 EXOR B, T A, H		
0576	CJMP SPOS * 2		%SET BIT 17 IF NEG. RESULT
0577	LOGM OR B, B17 A, SP D, SP		

0600	LOGM CRR1 ADIR A, H D, SH				
0601	LOGM EXOR B, B16 A, SP D, T				%H → SH, READ LEAST S. M.
0602	LOGL ADIR A, H D, SH				%INVERT BIAS BIT
0603	LOGL BDIR A, D B, Z				
0604	LOGM BDIR A, A B, Z				%D → A-LATCH, 0 → AC(L)
0605	LOOP BDI PLUSA B, HAC SH32 ASH0				
	SHR TGAC0 SACO				
0606	CJMP GREM MCRY1	616			%MULTIPLY LOOP
0607	CJMP ZERO ZTAD	567			%JMP IF CARRY
0610	LOGL BDIR B, AC D, D				%JMP IF ZERO
0611	LOGM BDIR B, AC D, A				%AC(L) → D
0612	ARM BM1 B, T D, T				%AC → A
0613	LOGM*AND B, B1 A, S				%T - 1 → T
0614	CJMP ZERO FETCH				%TEST TG
0615	LOGM CFC OR B, B0 A, D D, D	101			%JMP IF TG = 0
0616	LOGL BDIR B, HAC D, D				%SET BIT 0 → 1
0617	LOGM BDIR B, HAC D, A				%1/2 AC(L) → D
0620	JMP TTGN	613			%1/2 AC → A
0621	LOGM CR BDIR B, B5 D, SC				%40 → SC, READ EXPO.
0622	LOGL BDIR B, B0 D, SH				%SET SH(L) → 1, DIV ALGOR.
0623	ARM BMINA B, T A, H D, SP				%T - H → SP
0624	LOGM ANDCB B, B17 A, SP D, SP				%RESET BIT 17 IN SP
0625	LOGM CRR1 EXOR A, H B, T				%TEST SIGN, READ MOST S.M.
0626	CJMP SPOS *2				
0627	LOGM OR B, B17 A, SP D, SP				%SET BIT 17 → 1, NEGATIVE
0630	LOGM CRR1 ADIR A, H D, SCR				%H → SCR, READ LEAST S.M.
0631	CJMP SPOS FDVO	566			%OVERF. IF ZERO
0632	LOGM EXOR B, B16 A, SP D, T				%INVERT BIAS BIT
0633	LOGL BDIR B, D				%D → AC(L)
0634	ARM BDI B, A				%A → AC
0635	CJMP SPOS ZTAD	567			%FINISHED IF ZERO OR NOT NORM.
0636	LOGL BDIR B, HAC A, H				%1/2 AC(L) → AC(L), H → A-LATCH
0637	LOGM BDIR B, HAC A SCR				
0640	LOOP PLUS BMAA B, 2AC SACO SH32				
	ASH0 ENDID TGS0				%FDV LOOP
	FDVLO,				

* CFC is added in N12/42

* CFC is added in N12/42

[illegible]

0721	CARM BMINA A, D B, Z D, D NEG	%INVERT D IF NEG
0722	LOGM A, SS ADIR	%SIGN TEST
0723	CARM NEG BMINA A, A B, Z D, A	%COMPL. IF NEG
0724	LOGM EXOR B, A A, SS	%OVERFLOW TEST
0725	CJMP SNEG FETCZ	
0726	ARM CFC	
0727		
0727		
0727		
0727	LOGM AND B, B7 A, H	%TEST LOAD, STORE BLOCK
0727	CJMP ZERO SRB	
0730		
0731		
0731		
0731		
0731	ARM CPTR BM1 B, X D, P	% X - 1 → CP, CP → R
0731	LOGM CRR1 ADIR A, SP D, P	%SP → CP
0732	LOGM ADIR A, H D, SH	%CP → SH
0733	ILOGM BDIR B, SH D, SP ORIN DSPL CRR1	%LOAD SP
0734	LOGM ADIR A, H D, SH	%X → SH
0735	ILOGM BDIR B, SH D, X ORIN DSPL CRR1	%LOAD X
0736	LOGM ADIR A, H D, SH	%T → SH
0737	ILOGM BDIR B, SH D, T ORIN DSPL CRR1	%LOAD T
0740	LOGM ADIR A, H D, SH	%A → SH
0741	ILOGM BDIR B, SH D, A ORIN DSPL CRR1	%LOAD A
0742	LOGM ADIR A, H D, SH	%D → SH
0743	ILOGM BDIR B, SH D, D ORIN DSPL CRR1	%LOAD D
0744	LOGM ADIR A, H D, SH	%L → SH
0745	ILOGM BDIR B, SH D, L ORIN DSPL CRR1	%LOAD L
0746	LOGM ADIR A, H D, SH	%S → SH
0747	ILOGM BDIR B, SH D, S ORIN DSPL CRR1	%LOAD S
0750	LOGM ADIR A, H D, SH	%B → SH
0751	ILOGM BDIR B, SH D, B ORIN DSPL CFC	%LOAD B, FETCH
0752		
0753		
0753	%MULTIPLY CONTINUED	
0753	%ONE OPERAND IN A-REG	
0753	%SECOND OPERAND, CONTENT OF EFFECTIVE ADDRESS	
0753		

0753	MPYC,	LOGM ADIR A, A D, SCR CR	%A → SCR, READ SEC. OPL.
0754		CARM NEG BMNA A, A B, Z D, SCR	%INVERT IF NEG
0755		LOGM ANDCB B, B4 A, S D, S	%RESET DYNAMIC OVERFL.
0756		LOGM ADIR A, H D, SH	%SECOND OPER. → SH
0757		CARM NEG BMNA B, Z A, H D, SH	%INVERT IF NEG
0760		LOGM BDIR A, Z B, Z	%ZERO → AC AND A-LATCH
0761		LOGL BDIR A, SCR B, Z	%ZERO → AC(L) AND A → A-LATCH
0762		LOOP BDI PLUSA B, 2AC ASH31	%MULTIPLY LOOP
0763		LOGM BDIR B, 2AC	%TEST OVERF
0764		CJMP ZERO *3	%JMP IF NOT OVERFL.
0765		LOGM OR B, B4 A, S D, S	%SET STATIC OVERFL
0766		LOGM OR B, B5 A, S D, S	%SET DYNAMIC OVERFL
0767		LOGM EXOR A, H B, A	%TEST SIGN
0770		LOGL BDIR B, AC D, A	%RESULT → A
0771		CARM NEG CFC BMNA A, A B, Z D, A	%INVERT IF NEG.
0772			
0772			
0772	WAITC,	LOGM ADIR A, H D, SH	%PID → SH
0772		ARM A, IO B, PIM	%DECODED PIL → H
0773		LOGM AND CA A, H B, SH D, SCR	%CLEAR BIT IN PID
0774		ARM A, SCR B, PID D, IO	%SET PID
0775		JMP FETCH	
0776		0	101
0777			
1000			
1000			
1000			
1000	PANIN,	LOGM ADIR A, P D, SP	%ENTRY PANEL INTERRUPT
1000		JMP PANINC	
1001			1226
1002			
1002			
1002			
1002			
1002			
1002			
1002			
1002	BSBAC,	LOGM AND A, S B, B2	%TEST K
1002			

1003	CJMP NZERO BBSH	361	%0 → BIT, FETCH
1004	LOGM CFC ANDCB B, BM ORBW		%TEST K
1005	LOGM AND A, S B, B2	154	%JMP IF K = 1
1006	CJMP NZERO KONE2		%1 → SPECIFIED BIT
1007	LOGM OR B, BM ORBW		%1 → K, FETCH
1010	LOGM CFC OR B, B2 A, S D, S		%TEST K
1011	LOGM AND A, S B, B2	1037	%JMP IF K = 1
1012	CJMP NZERO KONE3		%0 → SPECIFIED BIT
1013	LOGM ANDCB B, BM ORBW		%0 → K, FETCH
1014	LOGM CFC ANDCB B, B2 A, S D, S		%TEST BIT
1015	LOGM AND ORBWO B, BM	1022	%JMP IF BIT = 1
1016	CJMP NZERO BONE1		%1 → K, FETCH
1017	LOGM CFC OR B, B2 A, S D, S	1017	%TEST BIT
1020	LOGM AND ORBWO B, BM		%JMP IF B = 1
1021	CJMP NZERO BLDCC +2		%0 → K, FETCH
1022	LOGM CFC ANDCB B, B2 A, S D, S		%TEST BIT
1023	LOGM AND ORBWO B, BM	101	%0 → K, FETCH
1024	CJMP NZERO FETCH		%TEST BIT
1025	LOGM CFC ANDCB B, B2 A, S D, S	1025	%JMP IF BIT = 1
1026	LOGM AND ORBWO B, BM		%K → K, FETCH
1027	CJMP NZERO BANDC +2		%TEST BIT
1030	LOGM CFC	1036	%JMP IF BIT = 1
1031	LOGM AND ORBWO B, BM		%K → K, FETCH
1032	CJMP NZERO BORCC +2		%TEST BIT
1033	LOGM CFC	1033	%JMP IF BIT = 1
1034	LOGM AND ORBWO B, BM		%1 → K, FETCH
1035	CJMP NZERO BORAC +2		%1 → SPECIFIED BIT
1036	LOGM CFC OR B, B2 A, S D, S		
1037			
1037			
1037	KONE3, LOGM OR B, BM ORBW		

1040	LOGM CFC ANDCB B, B2 A, S D, S	%C → K, FETCH
1041		
1041		
1041	CIIPP, ARM A, H B, MIS D, IO	%ION, IOF, PON, POF
1041	ARM CFC	
1042		
1043		
1043		
1043	% MOPC	
1043	% OPERATORS COMMUNICATION FOR NORD-10	
1043	MOPCM, ARM A, Z B, PIE D, IO	%MASTER CLEAR, RESET PIE
1044	ARM CHLEV	%CHANGE LEVEL
1045	LOGM OR A, Z B, B2 D, SCR	%SET BIT 2 IN SCR → 1
1046	ARM A, SCR B, PCR D, IO	%SET BANKNUMBER → 0
1047	ARM BDI ADD1 B, B2 D, SCR	%5 → SCR
1050	ARM A, SCR B, MIS D, IO	%RESET PAGING AND INTERRUPT
1051	LOGM BDIR B, B5 D, P	%BIT5 → CP
1052	ARM A, P B, MIS D, IO	%SET MOPC-BIT, BIT 5 MISC
1053	JMP MOPCR	
1054	LOGM ADIR A, P D, SP	%CP → SP
1055	LOGM BDIR B, B5 D, P	%BIT5 → CP
1056	ARM A, P B, MIS D, IO	%BIT-5 → MISC
1057	ILOGM ADIR A, A D, SS LE11 DSPL	%SAVE A-REG
1060		%BIT TO RESET LOAD LIGHT
1060	MOPCR, LOGM BDIR B, B15 D, SCR	
1061	LOGM A, SCR B, PAC D, IO	
1062	LOGM BDIR B, B5 D, SCR	
1063	LOGM BDIR B, B4 D, SH	
1064	IARM PLUS A, SCR B, SH D, SCR LE10 DSPL	%60 → SCR LEV10
1065	LOGM BDIR B, B3 D, SCR	
1066	ARM PLUS A, SCR B, B1 D, SH	
1067	ILOGM BDIR B, SH D, SCR LE6 DSPL	%12 → SCR LEV 6

1070	LOGM ADIR A, Z D, SS		%MPC STATUS, 0 → (SS LEV0)
1071	ILOGM ADIR A, Z D, SS LE12 DSPL		%CLEAR EXAMINE ADDRESS
1072	LOGM BDIR D, SCR B, B6		%100 → SCR
1073	ARM PLUS B, B7 A, SCR D, SCR		%SET BIT 7
1074	ILOGM ADIR A, SCR D, SS LE17 DSPL		%300 → (SS LEV17) CONSOLE DEV NR.
1075			
1076	NYFAF, ARM A, IO B, MPC	1756	%SAVE RETUR ADD.
1077	JMP ACT		
1100	REAC, ARM A, IO B, MPC	1176	%SAVE MPC
	JMP ASS8		
1101	%TEST TERMINATING CHARACTER IN A-REG		
1101	%OCTAL NUMBER IN SH IF BIT14 SS IS ONE		
1101	LOGM ADIR A, A		%IGNORE BLANK'S
1102	JMP TO REAC IF ZERO	1077	%100 → SCR
1103	LOGM BDIR B, B6 D, SCR		%100-A → A
1104	ARM BMINA A, SCR B, A D, A		%CHARACTER "100" READ
1105	CJMP ZERO MOPCM	1043	
1106	LOGM BDIR B, B3 D, SCR		
1107	ARM BMAMI B, A A, SCR		
1110	CJMP ZERO IEX	1436	%111, CHARACTER "1"
1111	ARM BMAMI B, AC A, SCR D, SCR		
1112	CJMP ZERO REX	1443	%122, CHARACTER "R"
1113	ARM PLUS B, B4 A, SCR		%102, B
1114	CJMP ZERO BANK	1373	%A + 20 + 1 → A
1115	ARM PLUS A, A B, B4 D, A ADDI	1273	%CHARACTER"/"
1116	CJMP ZERO EXAM		%TEST, 52*
1117	ARM PLUS A, A B, B2 ADDI D, A	1441	%A + 10 → A
1120	CJMP ZERO CLC		%CHAR. "&"
1121	ARM PLUS A, A B, B2 D, A	1504	%A + 2 → A
1122	CJMP ZERO ETSGN		%CHAR. "g"
1123	ARM PLUS A, A B, B1 D, A	1501	%A + 3 → A
1124	CJMP ZERO DOLL		%CHAR. "i"
1125	ARM PLUS A, A B, B1 ADDI D, A	1310	
1126	CJMP ZERO STPR		
1127	ARM BDI ADDI D, A B, A		

1130	CJMP ZERO REAC	1077	%SPACE
1131	ARM PLUS A, A B, B4 D, A		-- + 20 → A
1132	ARM PLUS A, A B, B1 ADD1		%A + 3 → A
1133	CJMP ZERO CHCR	1343	%CARRIAGE RET.
1134			
1134	LOGM BDIR D, SCR B, B6		%300 → SCR
1134	ARM PLUS B, B7 A, SCR D, SCR		%300 → SS(LEV17) CONSOLE DEVICE
1135	ILOGM ADIR A, SCR D, SS LE17 DSPL		%77 → A
1136	ARM BM1 B, B6 D, A		%RESET LOAD BIT
1137	LOGM ANDCB B, B17 D, SS		%SAVE MPC
1140	ARM A, IO B, MPC		%PRINT "?"
1141	JMP OUTCH	1735	%SET BIT 14
1142	LOGM BDIR B, B14 D, SCR		%SET ERROR INDICATOR
1143	ARM A, SCR B, PAC D, IO		%READ NEXT CHAR.
1144	ARM A, SCR B, PAC D, IO		
1145	JMP MOPCR	1060	
1146			
1146	%ROUTINE TO PRINT OCTAL NUMBERS		
1146	%PRINTS ON OPCOMDEV		%SAVE RETUR
1146	%NUMBER IN SH-MOST SIGN.		%SHIFT LEFT SHRO 16
1146	OUT8, IARM PLUS ADD1 A, H LE16 D, SCR DSPL		%FIRST OCTAL NUMB → A-REG
1147	LOGM BDIR B, B0 D, SC		%ADD 60
1147	LOOP SHRO TSC0		%SAVE MPC
1150	LOGM BDIR B, SH D, A		%PRINT FIRST DIGIT
1151	LOGM AND B, B0 A, A D, A		%5 → SCR
1152	IARM PLUS A, SCR B, AC D, A LE10		%-SCR → SCR (LEV12)
1153	ARM A, IO B, MPC		%3 → SC
1154	JMP OUTCH	1735	%SHIFT 3 SHRO LEFT
1155	ARM PLUS A, Z ADD1 B, B2 D, SCR		%7 → SCR
1156	IARM BM1A D, SCR B, Z A, SCR LE12 DSPL		%SH · 7 → AC
1157	ARM BM1 B, B2 D, SC		%AC + 60 → AC
1160	LOOP SHRO TSC0		%SAVE MPC
1161	ARM BM1 B, B3 D, SCR		
1162	LOGM AND B, SH A, SCR		
1163	IARM PLUS A, SCR B, AC D, A LE10		
1164	ARM A, IO B, MPC		
1165			

1166	JMP OUTCH	1735	%PRINT ONE DIGIT
1167	ILOGM ADIR D, SCR A, SCR LE12		
1170	IARM PLUS ADD1 D, SCR B, Z A, SCR LE12		
	DSPL		%COUNT
1171	CJMP NZERO NECHP	1160	%40 → A
1172	LOGM BDIR D, A B, B5		
1173	LOGM ADIR B, MPC A, IO		
1174	CALL OUTCH	1735	%PRINT SPACE
1175	JMP RETU1	1223	
1176			
1176			
1176			
1176			
1176	%THIS ROUTINE READS AN OCTAL NUMBER		
1176	%RETURN:OCTAL NUMBER IN SH(MOSTS)		
1176	%TERMINATING CHARACTER IN A-REG		
1176			
1176	ASS8, IARM PLUS ADD1 A, H B, Z D, SCR LE16		%SAVE RETURN
	DSPL		%0 → SH
1177	LOGM ADIR A, Z D, SH		%RESET OCTAL FLAG
1200	LOGM ANDCB B, B14 A, SS D, SS		%SAVE RETURN
1201	ARM A, IO B, MPC	1716	%READ ONE CHARACTER
1202	JMP INCH		%RESET PARITY
1203	LOGM ANDCB D, P B, B7 A, A		%AC - 60 → A
1204	IARM BMINA D, A B, AC A, SCR LE10	1220	%JMP IF NOT OCT. DIGIT
1205	CJMP SNEG EASS8		%10 → SCR
1206	LOGM BDIR B, B3 D, SCR		%A-10 → A
1207	ARM BMINA B, A A, SCR D, A	1220	%JMP IF NOT OCT. DIGIT
1210	CJMP SPOS EASS8		%7 → A
1211	ARM BM1 B, B3 D, A		%CP · A → SCR
1212	LOGM AND A, P B, A D, SCR		%3 → SC
1213	ARM BM1 B, B2 D, SC		%SHIFT 3 LEFT
1214	LOOP TSC0		%NEW OCTAL DIGIT → SH
1215	LOGM OR A, SCR B, SH D, SH		%SET OCTAL FLAG
1216	LOGM OR A, SS B, B14 D, SS		%READ NEXT CHAR.
1217	JMP RLOOP	1201	

1220	EASS8,	LOGM A DIR D, A A, P			
1221		IARM BMINA B, AC A, SCR LE6			
1222		JMP TO RLOOP IF ZERO	1201		%TERMINATION CHAR. FROM P
1223	RETU1,	ILOGM A DIR A, SCR D, P LE16			%TEST FOR LF
1224	RETU,	LOGM A, P B, CAR D, IO			%LINE FEED IGNORE
1225		JMP, CAR			%RETUR ADD. TO SCR
1226					%SCR → CAR
1227	PANINC,	LOGM B DIR B, B5 D, P			%BIT 5 SET IN CP
1228		ARM A, P B, MIS D, IO			%SET MOPC-BIT 5 IN MISC
1229		ARM A, IO B, OPR			%READ OPR
1230		ILOGM A DIR D, SS A, H LE12 DSPL			%SAVE OPR
1231		ARM A, IO B, PAS			%READ PANEL STATUS
1232		LOGM A DIR A, H D, P			%SET STATUS FOR PANEL INTERRUPT
1233		LOGM B DIR B, B6 D, SS	1257		
1234		JMP PANT2			
1235					
1236					
1237					
1238					
1239					
1240					
1241					
1242					
1243					
1244					
%THIS ROUTINE TESTS PANEL BUTTONS AND					
1236	PANTT,	LOGM AND B, B17 A, SS			%TEST LOAD
1237		CJMP NZERO RPANT	1717		%JMP TO INCH
1238		ILOGM B DIR LE2 B, SH D, SS DSPL			%SAVE SH-REG
1239		ARM A, IO B, PAS			%PANEL STATUS → H
1240		LOGM A DIR A, H D, P			%SAVE PAS
1241		ILOGM A DIR A, SS D, A LE11			%UNSAVE A
1242		LOGM AND B, B14 A, H			%TEST SET ADDRESS

1245	CJMP NZERO SETAD	1446	%TEST RESTART
1246	LOGM AND B, B12 A, H		%JMP TO RESTART
1247	CJMP NZERO RESTA	1304	%TEST DEPOSIT
1250	LOGM AND B, B13 A, H	1467	%DEPOSIT, ADDR, IN SS LE12
1251	CJMP NZERO DEPP		%TEST CONTINUE
1252	LOGM AND B, B11 A, H	1307	%START PROG. IN MAIN MEM.
1253	CJMP NZERO STFP		%TEST LOAD
1254	LOGM AND B, B10 A, H	1503	%LOAD FROM LOAD DEV.
1255	CJMP NZERO LOAD	1256	%JMP TO REGEX. OR MEM. EX
1256	JMP PANT1		
1256			
1256			
1257			
1260		1462	%SAVE SH
1261			%TEST NOOP
1262			
1263			%RESET BIT IN PAS
1264			%SET PAC
1265		1456	%TEST REG. OR MEM. EXAM.
1266			%SET CAR
1267			%READ REG
1270			%SET LAMP DISP.
1271		1462	
1272			
1273			
1273			
1273			
1273			
1273			
1273			
1273			
1274			
1275			
1276			
1277			

PANT1,	ILOGM BDIR B, SH D, SS LE2 DSPL		
PANT2,	LOGM AND B, B15 A, P		
	CJMP NZERO RETPA		
	ARM BM1 B, B7 D, SH		
	LOGM AND B, SH A, P D, SCR		
	ARM A, SCR B, PAC D, IO		
	LOGM AND B, B7 A, P		
	CJMP NZERO MEXM		
	ARM A, P B, CAR D, IO		
	ARM		
	ILOGM ADIR ORBWO D, SCR		
	ARM A, SCR B, LMP D, IO		
	JMP RETPA		

	%JMP TO THIS ROUTINE WHEN "/" IS TYPED		
EXAM,	LOGM OR B, B10 A, SS D, SS		
	LOGM AND B, B7 A, SS		
	CJMP NZERO REXAM		
	LOGM ANDCB B, B16 A, SS D, SS		
	ILOGM BDIR B, SH D, SS LE12 DSPL	1321	%SET EXAM BIT
			%TEST REGISTER EXAM
			%JMP REG EXAM IF NOT ZERO
			%RESET REG DEP
			%SAVE ADDRESS

```

1300 EXRO, ILOGM ADIR A, SS D, P LE12 %ADDRESS → CP
1301 ARM CPTR BM1 B, P D, P %CP - 1 → CP
1302 ARM CRR1
1303 JMP IEXA1 1424
1304
1304 %START, <OCTALN.>:
1304 %START ADDRESS IN SH IF OCTALNUMBER WRITTEN ELSE IN
1304 %SP (CONTINUE)
1304
1304 RESTA, LOGM BDIR B, B4 D, SP %SET RESTART ADDRESS
1305 STSP, ARM A, Z B, PIE D, IO %RESET PIE
1306 ARM CHLEV %CHANGE LEVEL
1307 LOGM ADIR A, Z D, SS %0 → SS, 0 → OCTAL FLAG
1310 LOGM AND B, B14 S, SS D, SS %PREPARE OCTAL TEST
1311 ILOGM ADIR A, SS D, A LE11 %UNSAVE A-REG
1312 ARM A, Z B, MIS D, IO %0 → MISC
1313 ILOGM ADIR A, SS LE0 %TEST OCTAL FLAG
1314 CLOGM NZERO BDIR B, SH D, SP %START ADDRESS → SP
1315 LOGM BDIR B, B15 D, SCR %SET BIT FOR RESET LOAD
1316 LOGM OR B, B11 A, SCR D, SCR %SET CONTINUE BIT
1317 ARM A, SCR B, PAC D, IO %SET BITS TO PANEL CONT.
1320 LOGM ADIR A, SP D, P CFC %START ADDRESS → CP, START
1321
1321 %REGISTER EXAMIN. JMP FROM EXAM
1321 %TESTS ON BIT 15 IN STATUS(SS) AND JUMPS
1321 %TO IEXAM IF "INTERNAL REG"
1321
1321 REXAM, LOGM ANDCB B, B7 A, SS %RESET REG EXAM
1322 LOGM OR B, B16 A, SS D, SS %SET REG DEP
1323 ILOGM BDIR B, SH D, SS LE7 DSPL %SAVE REG NUMBER
1324 LOGM AND A, SS B, B15 %TEST INTERNAL REG

```

[illegible]

1365	%INTERNAL REGISTER DEPOSIT	
1365		
1365		
1365		
1365	IRD,	%READ REG NUMB.
1365	ILOGM ADIR A, SS D, SCR LE17	%SET REG. NUMBER IN CAR
1366	ARM A, SCR B, CAR D, IO	%NUMBER WRITTEN IN SCR
1367	LOGM BDIR B, SH D, SCR	%DEPOSIT IN SPESY. REG
1370	ARM ADIR A, SCR B, IR3 D, IO	%RESET INTERNAL REG BIT
1371	LOGM ANDCB A, SS B, B15 D, SS	
1372	JMP NRDP	1406
1373		
1373	BANK,	
1373	LOGM BDIR B, SH D, SCR	%SET BIT 2 IN SCR → 1
1374	LOGM OR B, B2 A, SCR D, SCR	%SET BANKNUMBER IN PCR
1375	ARM A, SCR B, PCR D, IO	%RESET BIT FOR R-I
1376	LOGM ANDCB B, B7 A, SS D, SS	
1377	JMP NRDP1	1410
1400		
1400		
1400	%**==**==*	
1400		
1400		
1400	JMP 400	%INTERRUPT AND PANEL INT
1401	JMP MOPCM	%MASTER CLEAR
1402	JMP MOPC	%STOP
1403		
1403		
1403	%**==**==*	
1403		
1403	REDEP,	%READ NUMBER
1403	ILOGM ADIR A, SCR D, SH LE1	%DEPOSIT IN REG
1404	RPDEP,	%SAVE A-REG
1404	ILOGM BDIR ORBW DSPL B, SH	%RESET DEPOSIT BIT
1405	ILOGM ADIR A, A D, SS LE11 DSPL	%RESTORE REGEX BIT
1406	LOGM ANDCB B, B11 A, SS D, SS	
1407	LOGM ANDCB A, SS B, B16 D, SS	

NRDP1,	LOGM ANDCB A, SS B, B10 D, SS				
	JMP TO REAC				1077
%ROUTINE FOR REGISTER DEPOSIT					
REGDP,	LOGM AND B, B14 A, SS			%TEST OCTAL NUMBER WRITTEN	
	CJMP ZERO NRDP			%NO DEPOSIT	1406
	LOGM AND B, B15 A, SS			%TEST "INTERNAL REG"	
	CJMP NZERO IRD			%INTERNAL REGISTER DEPOSIT	1365
	ILOGM BDIR B, SH D, SCR LE1 DSPL			%SAVE NUMBER	
	LOGM OR B, B11 A, SS D, SS			%SET BIT FOR DEPOSIT	
	JMP ERDP			%JMP TO GEN, REGNR. AND LEVEL	1326
% "INTERNAL REG" EXAMIN					
IEXAM,	LOGM BDIR B, SH D, SCR			%REG. NUMBER → SCR	
	ARM A, SCR B, CAR D, IO			%SET REG. NUMBER → CAR	
	ARM A, IO B, IR 3			%READ REG CONTENT	
IEXA1,	LOGM ADIR A, H D, SH			%SAVE RETUR	
IEXA,	ARM A, IO B, MPC			%PRINT NUMBER	1146
	JMP OUT8			%READ NEXT INP	1077
	JMP REAC				
PRLF,	IARM PLUS A, H B, Z ADD1 D, SCR LE5			%SAVE RETUR	
	DSPL				

1431	ILOGM ADIR A, SCR LE6 D, A	%12 → A
1432	ARM A, IO B, MPC	%SAVE RETUR
1433	JMP OUTCH	1735 %PRINT LF
1434	RETU5, ILOGM ADIR A, SCR D, P LE5	%RETUR ADR → CP
1435	JMP RETU	1224
1436		
1436		
1436		
1436		
1436		
1436	%CHARACTER "I" WRITTEN	
1436		
1436	IEX, LOGM OR B, B15 A, SS D, SS	%SET FLAG FOR I
1437	LOGM OR B, B7 A, SS D, SS	%SET FLAG FOR R + I
1440	JMP NRDP1	1410
1441		
1441	CLC, ILOGM ADIR A, SS D, SH LE12	%READ CURRENT LOC. COUNTER
1442	JMP IEXA	1425
1443		
1443	%CHARACTER "R" WRITTEN	
1443		
1443	REX, ILOGM BDIR B, SH D, SS LE14 DSPL	%SAVE LEVEL
1444	LOGM ANDCB B, B15 A, SS D, SS	%RESET BIT FOR I
1445	JMP IEX + 1	1437
1446		
1446		
1446	SETAD, ARM A, IO B, OPR	%READ SWITCH REGISTER
1447	ILOGM ADIR A, H D, SS LE12 DSPL	%SAVE ADDRESS
1450	LOGM OR B, B0 A, P D, SH	%SET BIT 0 (BIT 2 PCR)
1451	LOGM BDIR B, B1 D, SC	%SET SHIFT COUNTER
1452	LOOP SHRO TSC0	%SHIFT BANK NUMBER
1453	LOGM BDIR B, SH D, SCR	

1454
1455
1456
1456
1456
1456
1457
1460
1461
1462
1463
1464
1465
1466
1467
1467
1467
1467
1467
1467
1467
1470
1471
1472
1473
1474
1475
1476
1477
1477
1500
1500
1500

1501	%ENTRY IN MOPC AFTER \$, & OR LOAD COMMAND		
1501	%(SH1)= ALD-NUMBER, SPECIFIED ON CONSOLE DEVICE		
1501	%IF NO ALD NUMBER IS SPECIFIED, THE DEFAULT NUMBER IS READ FROM ALD		
1501	%ALD NUMBER FORMAT		
1501	%BIT 15 : EXTN, EXTENDED LOAD FUNCTION		
1501	%BIT 14 : RESTART		
1501	%BIT 13 : MASS STORAGE LOAD		
1501	%BIT 12 : OTAL LOAD		
1501			
1501	DOLL, LOGM BDIR D, A B, B14	%10000 → A, AFTER \$	1505
1502	JMP DOLET		
1503	LOAD, LOGM BDIR B, B17 D, SS	%SET LOAD-FLAG, RESET OCTAL READ FLAG	
1504			
1504	ETSGN, LOGM BDIR D, A B, Z	%0 → A, AFTER & OR LOAD	
1505			
1505	DOLET, LOGM ADIR B, PAS A, IO	%READ PANEL	
1506	ARM BM1 D, P B, B10	%377 → P	
1507	LOGM AND D, P B, P A, H		
1510	LOGM OR D, P B, B10 A, P	%SET LOAD BIT IN PANEL CONTROL	
1511	LOGM ADIR D, IO B, PAC A, P		
1512	LOGM AND B, B14 A, SS	%TEST IF OCTAL NUMBER READ	
1513	JMP TO DE0 IF NZERO	%JMP OCTAL READ	1516
1514	LOGM ADIR B, ALD A, IO	%GET ALD	
1515	LOGM ADIR D, SH A, H		
1516	LOGM OR D, A A, A B, SH	%OR IN B14 IF \$ WRITTEN	
1517	LOGM AND B, B17 A, A		
1520	JMP TO EXTN IF NZERO	%EXTENDED LOAD FUNCTION	1574
1521			
1521	LOGM AND B, B16 A, A	%RESTART FUNCTION?	
1522	JMP TO RSTRT IF NZERO	%YES	1576
1523	LOGM AND B, B15 A, A		
1524	JMP TO MASS IF NZERO	%MASS STORAGE LOAD	1601
1525			
1525	DE1, ILOGM ADIR D, SS A, A LE17 DSPL	%DEV. NR. → SS LEV17	

LOGM BDIR D, SS B, B17			
LOGM OR D, SS B, B10 A, SS			%SET LOAD FLAG
LOGM AND B, B14 A, A			%SET EXAMINE BIT IN SS
JMP TO NYFAF IF NZERO		1075	%OCTAL LOAD IF A = 010000
JMP TO BINL		1532	
%BINARY LOAD			
%WILL PICK UP NECESSRY PARAMETERS FROM A STANDARD (OCTAL)			
%BOOTSTRAP AND LOAD THE BINARY INFORMATION INTO CORE			
%IMMEDIATELY FOLLOWING THE CHECKSUM THERE IS AN ACTION			
%COMMAND. IF TERMINATING			
%CHARACTER IS A BLANK, THEN THE LOADED PROGRAM IS STARTED IN			
%THE START ADDRESS FOUND IN THE BOOTSTRAP. OTHERWISE THE COMMAND			
%IS ACTED UPON BY MOPC			
BINL,	LOGM ADIR B, MPC A, IO		
	CALL ACT	1756	%ACTIVATE DEVICE
	LOGM ADIR D, SH A, Z		%0 → SH
SEEK,	LOGM BDIR D, SP B, SH		%SH → SP (POSSIBLE START ADDRESS)
SIKI,	LOGM ADIR B, MPC A, IO		
	CALL ASS8	1176	%OCTAL NUMBER READ?
	LOGM AND B, B14 A, SS		%NO
	JMP TO SIKI IF ZERO	1536	% A - 40 → AC
	ARM BMINA B, B5 A, A		%AC + 1 → AC
	ARM PLUS ADD1 B, AC A, Z		%SEEK FOR '1' AS TERMINATOR
	JMP TO SEEK IF NZERO	1535	
	LOGM ADIR B, MPC A, IO		
	CALL BIN	1645	

%EXTN

```

1576 %RSTRT
1576 %RESTART FUNCTION (NOT TO BE CONFUSED WITH THE RESTART BUTTON)
1576 %WHEN THE RSTRT-BIT (BIT 14) OF A IS SET, THE CPU IS STARTED
1576 %IN 4* (THE ADDRESS FOUND IN BIT0-13 IN A ). BIT 0-1 IN THE
1576 %RESULTING ADDRESS IS 0
1576
1576 RSTRT, ARM PLUS D, A B, A A, A %*2
1577 ARM PLUS D, SP B, A A, A %*4
1600 JMP TO STSP 1305 %START PROGRAM
1601
1601
1601 %MASS
1601 %MASS STORAGE LOAD
1601 %MASS WILL LOAD 1K WORDS FROM MASS STORAGE ADDRESS 0 INTO MEMORY
1601 %FROM MEMORY ADDRESS0
1601 %(LOWEST) MASS STORAGE REGISTER ADDRESS IS FOUND IN BIT 0-10 IN ALD
1601 %THE ACTUAL MASS STORAGE MUST CONFORM WITH THE DRUM/DISC
1601 %PROGRAMMING SPECIFICATION.
1601
1601 MASS, ARM PLUS D, P B, B0 A, A CPTR %DEV NR(DNO) + 1 → CP, CP → R
1602
1602 MAS1, LOGM ADIR D, A A, Z %0 → A
1603 LOGM ADIR D, SCR A, R %DNO + 1
1604 LOGM ADIR B, MPC A, IO
1605 CALL IOXR 1632 %0 → CORE ADDRESS
1606 ARM PLUS D, SCR B, B1 A, R %DNO + 3
1607 LOGM ADIR B, MPC A, IO
1610 CALL IOXR 1632 %0 → BLOCK ADDRESS

```

[illegible]

1640	LOGM ADIR D, IO B, CAR A, SCR		%SET UP CAR
1641	LOGM ADIR D, IO B, IO A, A		%OUT -
1642	LOGM ADIR B, IO A, IO		%- IN -
1643	LOGM ADIR D, A A, H		% - TO A-REG
1644	JMP RETU	1224	%STANDARD RETURN
1645			
1645			
1645			
1645			
1645	%BIN		
1645	%BIN READS A 16BIT WORD IN TWO BYTES		
1645	%CALL WITH : DEVICE-NO IN SS17		
1645	%RETURNS WITH : DATA IN SH1		
1645			
1645			
1645	BIN, IARM PLUS ADD1 D, SCR B, Z A, H LE5 DSPL		%SAVE RETURN
1645			
1646	BIN1, LOGM ADIR B, MPC A, IO		
1647	CALL INCH	1716	%READ 1, BYTE
1650	LOGM ADIR D, SH A, A		
1651	LOGM BDIR D, SC B, B3		%8 → SC
1652	LOOP		%SHIFT 8 LEFT
1653			
1653	BIN2 LOGM ADIR B, MPC A, IO		
1654	CALL INCH	1716	%READ 2, BYTE
1655	LOGM OR D, SH B, SH A, A		%JOIN
1656	JMP RETU 5	1434	%STANDARD RETURN
1657			
1657			
1657	%MAIN MEMORY CHECK		
1657	%MICRO-PROGRAM, VERSION C		
1657	%NJL 6/6/73		

```

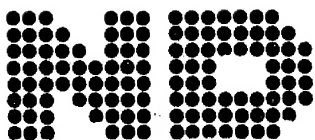
1657 %
1657 %TESTS MEMORY FROM ADDRESS SPECIFIED IN B-REG AND UP TO
1657 %ADDRESS SPECIFIED IN X-REG
1657 %
1657 %ERROR INDICATION:
1657 %   T=FAILING BITS      [OR6 ]
1657 %   P=FAILING ADDRESS  [OR2 ]
1657 %   D=ERROR PATTERN    [OR1 ]
1657 %   L=TEST PATTERN     [OR4 ]
1657 %   B=START ADDRESS    [OR3 ]
1657 %   X=STOP ADDRESS     [OR7 ]
1657 %
1657 %IF NO ERROR IS FOUND T = 000000
1657 %
1657 %PATTERNS USED :
1657 %
1657 %   000001
1657 %   000002
1657 %   000004
1657 %   .....
1657 %   .....
1657 %   100000
1657 %
1657 %
1657 % ADDRESS STORED IN ADDRESS (THIS PATTERN IS RUN 16 TIMES)
1657 %
1657 % AND THEIR COMPLEMENTS
1657 %
1657 %USE OF REGISTERS IN THIS PROGRAM :
1657 %
1657 %   A = 000000 USE T AS TEST DATA
1657 %   A = 177777 USE ADDRESS AS TEST DATA
1657 %
1657 %   SCR = 000000 WRITE TEST DATA INTO CORE
1657 %   SCR = 177777 READ TEST DATA FROM CORE
1657 %
1657 %   D = 000000 USE TEST DATA AS IS
1657 %   D = 177777 USE COMPLEMENT OF TEST DATA
1657 %
1657 %   L =          TEST DATA

```

		%0 → SS	INIT PHASE
MMCC,	LOGM ADIR D, A A, Z		INIT PHASE
MM00,	LOGM ADIR D, D A, Z	%0 → D	INIT POLARITY
MM0,	LOGM BDIR D, T B, B0	%1 → T	INIT DATA
MM1,	LOGM ADIR D, SCR A, Z	%0 → SCR	INIT STORE
MM2,	ARM BMI D, P B, B CPTR	%B - 1 → CP, CP → R	INIT ADDRESS
MM3,	ARM PLUS ADD1 D, L B, Z A, R	%R + 1 → L	MAKE DATA
	LOGM ADIR A, A	%T → L IF A = 0	
	LOGM ADIR D, L A, T IF ZERO	%L → L IF D = 17777	
	LOGM EXOR D, L B, D A, L		
	LOGM ADIR A, SCR		
	JMP TO MM4 IF NZERO	%LOAD IF (SCR) = 0	
	ARM A, L CWR1	%STORE L IN (R) + 1	
	JMP TO MM41		
MM4,	ARM CRR1	%R + 1 → H	
	LOGM EXOR B, L A, H		
	JMP TO ERR IF NZERO	%ERROR DETECTED	
MM41,	ARM BMINA B, X A, R	%TEST FOR LAST LOCATION	
	JMP TO MM3 IF NZERO	%CONTINUE	
	LOGM ADIRC D, SCR A, SCR	%SWITCH LOAD/STORE	
	JMP TO MM2 IF NZERO		
	ARM PLUS D, T B, T A, T	%T + T → T	SWITCH DATA
	JMP TO MM1 IF NZERO		

			SWITCH POLARITY
1705	MM45,	LOGM ADIRC D, D A, D	
1706		JMP TO MM0 IF NZERO	
1707			1661 %DO COMPLEMENT
1707	MM5,	LOGM ADIRC D, A A, A	
1710		JMP TO MM00 IF NZERO	
1711		JMP TO MOPC	1661 %CONTINUE WITH ADDRESS IN ADDRESS
1712			1054 %END OF TEST
1712	ERR,	LOGM ADIR D, D A, H	
1713		LOGM ADIR D, SP A, R	%DATA READ
1714		LOGM EXOR D, T B, L A, D	%FAILING ADDRESS R → SP
1715		JMP QUM	%FAILING BITS → T
1716			1134 %MOPC ERROR ENTRY-POINT
1716			
1716	%INCH		
1716		%INPUT CHARACTER (BYTE) FROM DEVICE FOUND IN (SS17)	
1716		%RETURNS CHARACTER IN A0	
1716		***** SCR0 IS DESTROYED ***	
1716		%IF LOAD-BIT IS SET (BIT17 IN SS0) THEN THERE WILL BE NO ECHOING	
1716		%ON DEVICE-NUMBER (SS17) +4	
1716			
1716	INCH,	IARM PLUS ADD1 D, SCR B, Z A, H LE15 DSPL	
1717			
1717	RPANT,	ILOGM ADIR D, SCR A, SS LE17	%RETURN FROM PANTT) DN0 → SCR
1720		ARM PLUS D, SCR B, B1 A, SCR	%DN0 + 2 → SCR
1721		LOGM ADIR B, MPC A, IO	
1722		CALL IOXR	1632 %READ STATUS
1723		LOGM AND B, B3 A, A	%READY FOR TRANSFER ?
1724		JMP TO PANTT IF ZERO	1236 %NO, TEST PANEL
1725		ILOGM ADIR D, SCR A, SS LE17	%YES, DN0 → SCR
1726		LOGM ADIR B, MPC A, IO	
1727		CALL IOXR	1632 %READ CHARACTER

1730	ILOGM ADIR D, SCR A, A DSPL LE13	%SAVE A
1731	LOGM ADIR B, MPC A, IO	
1732	CALL ACT	1756 %ACTIVATE DEVICE
1733	ILOGM ADIR D, A A, SCR LE13	%UNSAVE A0
1734	JMP OUT1	1736 %FOR ECHO-ING
1735		
1735		
1735	%OUTCH	
1735	%WRITES A CHARACTER ON DEVICE (SS17) +4	
1735	**** SCR0 IS DESTROYED ***	
1735		
1735	OUTCH, IARM PLUS ADD1 D, SCR B, Z A, H LE15 DSPL	
1736		
1736	LOGM AND B, B17 A, SS	
1737	JMP TO OUT3 IF NZERO	1754 %NO OUTPUT IF LOAD MODE
1740	ILOGM ADIR D, SCR A, A LE13 DSPL	%SAVE A0
1741	IARM PLUS ADD1 D, SCR B, Z A, SS LE17	%DN0 + 1 → SCR
1742	ARM PLUS ADD1 D, SCR B, B2 A, SCR	%DN0 + 6 → SCR
1743		
1743	LOGM ADIR B, MPC A, IO	
1744	CALL IOXR	1632 %READ STATUS
1745	LOGM AND B, B3 A, A	%READY FOR TRANSFER?
1746	JMP TO OUT2 IF ZERO	%NO
1747	LOGM ADIR A, SCR	%SCR → AC
1750	ARM BM1 D, SCR B, AC	%DN0 + 5 → SCR
1751	ILOGM ADIR D, A A, SCR LE13	%UNSAVE A0
1752	LOGM ADIR B, MPC A, IO	
1753	CALL IOXR	1632 %WRITE CHARACTER
1754		
1754	ILOGM ADIR D, P A, SCR LE15	
1755	JMP RETU	1224 %STANDARD RETURN
1756		
1756	%ACTIVATES DEVICE	
1756		
1756	ACT, IARM PLUS ADD1 D, SCR B, Z A, SS LE17	%DN0 + 1 → SCR
1756		



NORSK DATA A.S.

Lørenveien 57 - Postboks 163, Økern

OSLO 1

COMMENT AND EVALUATION SHEET

NORD-10/S MICROPROGRAM
February 1978

Publ. No. ND-06.010.01

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM

– we make bits for the future

NORSK DATA A.S LØRENVEIEN 57 OSLO 5 NORWAY PHONE: 21 73 71 TELEX: 18284